2011

Camera Planning and Fusion in a Heterogeneous Camera Network

Jian Zhao
University of Kentucky, shiningsword@gmail.com

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained and attached hereto needed written permission statements(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine).

I hereby grant to The University of Kentucky and its agents the non-exclusive license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless a preapproved embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's dissertation including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Jian Zhao, Student

Dr. Sen-Ching S. Cheung, Major Professor

Dr. Zhi D. Chen, Director of Graduate Studies

ABSTRACT OF DISSERTATION

Jian Zhao

The Graduate School
University of Kentucky
2011

Camera Planning and Fusion in a Heterogeneous Camera Network

ABSTRACT OF DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for
the degree of Doctor of Philosophy
in the College of Engineering at the
University of Kentucky

By
Jian Zhao
Lexington, Kentucky

Director: Dr. Sen-Ching S. Cheung, Professor of Electrical and Computer
Engineering
Lexington, Kentucky 2011

ABSTRACT OF DISSERTATION

Camera Planning and Fusion in a Heterogeneous Camera Network

Wide-area camera networks are becoming more and more common. They have wide-range of commercial and military applications from video surveillance to smart home and from traffic monitoring to anti-terrorism. The design of such a camera network is a challenging problem due to the complexity of the environment, self and mutual occlusion of moving objects, diverse sensor properties and a myriad of performance metrics for different applications. In this dissertation, we consider two such challenges: camera planing and camera fusion. Camera planning is to determine the optimal number and placement of cameras for a target cost function. Camera fusion describes the task of combining images collected by heterogenous cameras in the network to extract information pertinent to a target application.

I tackle the camera planning problem by developing a new unified framework based on binary integer programming (BIP) to relate the network design parameters and the performance goals of a variety of camera network tasks. Most of the BIP formulations are NP hard problems and various approximate algorithms have been proposed in the literature. In this dissertation, I develop a comprehensive framework in comparing the entire spectrum of approximation algorithms from Greedy, Markov Chain Monte Carlo (MCMC) to various relaxation techniques. The key contribution is to provide not only a generic formulation of the camera planning problem but also novel approaches to adapt the formulation to powerful approximation schemes including Simulated Annealing (SA) and Semi-Definite Program (SDP). The accuracy, efficiency and scalability of each technique are analyzed and compared in depth. Extensive experimental results are provided to illustrate the strength and weakness of each method.

The second problem of heterogeneous camera fusion is a very complex problem. Information can be fused at different levels from pixel or voxel to semantic objects, with large variation in accuracy, communication and computation costs. My focus is on the geometric transformation of shapes between objects observed at different camera planes. This so-called the geometric fusion approach usually provides the most reliable fusion approach at the expense of high computation and communication costs. To tackle the complexity, a hierarchy of camera models with different levels of complexity was proposed to balance the effectiveness and efficiency of the camera

network operation. Then different calibration and registration methods are proposed for each camera model. At last, I provide two specific examples to demonstrate the effectiveness of the model: 1)a fusion system to improve the segmentation of human body in a camera network consisted of thermal and regular visible light cameras and 2) a view dependent rendering system by combining the information from depth and regular cameras to collecting the scene information and generating new views in real time.

KEYWORDS: Sensor Planning, Camera Placement, Sensor Fusion, Human Segmentation, Multi-camera Fusion

Author's signature: Jian Zhao

Date: October 18, 2011

Camera Planning and Fusion in a Heterogeneous Camera Network

By
Jian Zhao

Director of Dissertation: Sen-Ching S. Cheung

Director of Graduate Studies: Zhi D. Chen

Date: October 18, 2011

RULES FOR THE USE OF DISSERTATIONS

Unpublished dissertations submitted for the Doctor's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgments.

Extensive copying or publication of the dissertation in whole or in part also requires the consent of the Dean of the Graduate School of the University of Kentucky.

A library that borrows this dissertation for use by its patrons is expected to secure the signature of each user.

Name	Date

DISSERTATION

Jian Zhao

The Graduate School
University of Kentucky
2011

Camera Planning and Fusion in a Heterogeneous Camera Network

DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for
the degree of Doctor of Philosophy
in the College of Engineering at the
University of Kentucky

By
Jian Zhao
Lexington, Kentucky

Director: Dr. Sen-Ching S. Cheung, Professor of Electrical and Computer
Engineering
Lexington, Kentucky 2011

For my wife and parents

ACKNOWLEDGMENTS

I would never have been able to finish my dissertation without the guidance of my committee members, help from friends, and support from my family and wife.

I am heartily thankful to my supervisor, Dr. Sen-Ching Samson Cheung, whose encouragement, guidance and support from the initial to the final level enabled me to develop an understanding of the subject. I also want to give many thanks to my committee members, Dr. Yuming Zhang, Dr. Daniel Lau, Dr. Ruriko Yoshida and Dr. Ruigang Yang, for the time they spent on my thesis review. I appreciate all of the suggestions and comments. I also want to acknowledge my co-workers, Dr. Ruriko Yoshida, Dr. David Haws, Dr. Thinh Nguyen and Ju Shen who have contributed to my dissertation work.

I would also like to thank my parents. They were always supporting me and encouraging me with their best wishes. Finally, I would like to thank my wife, Jingjing Liu. She was always there cheering me up and stood by me through the good times and bad.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

For human beings, vision is one of the most important perceptions of outside world. Visual perception collects information from surroundings by the effect of visible light reaching the eyes. For centuries, scientists and engineers have been working on building sensors which can mimic human vision. Today, sensor technology has advanced far beyond the capacity of human vision: video cameras that can generate images similar to human vision; thermal cameras are used to obtain the surface temperature of an object; gamma and X-ray cameras are used in medical diagnosis. Recent advances in depth cameras also make the direct perception of 3D geometry more plausible. The dissertation generalizes the term *camera* as those sensors which generate images by collecting rays of electromagnetic (EM) radiation of all frequencies. Unlike antennas, cameras usually use much higher frequency($\geq$ 3 THz) EM radiations in order to limit the diffraction effect, which increases resolution.

At the same time, recent advances in sensor networking and distributed processing make it possible for cameras to work jointly on large scale applications. The widespread deployment of such networks with heterogeneous cameras poses an important design problem. Not only does it determine the coverage of surveillance, it also has a direct impact on the appearance of objects in the cameras which dictates the performance of all subsequent processes. Furthermore, due to the variety of sensors in the network, a distributed fusion algorithm has to be carefully designed to combine all the information available in order to achieve performance goals.

1.1 Camera Network

1.1.1 Introduction to Camera Network

A camera network refers to a network of distributed camera nodes [1]. Each camera node has its own processing capacity, but the network itself can perform much more sophisticated tasks by aggregating and fusing the information from individual sensors. The emergence of camera network has significant impact on a wide range of applications. The applications mentioned in this section are only a small portion of the camera network applications that directly relate to the research in this dissertation.

Video surveillance is a traditional application for camera network. Recent advances in camera networks help video surveillance in many ways. Camera planning algorithms can be useful in minimizing the cost of deploying surveillance systems. Local processing and sensor fusion techinques can help the system to automatically detect suspicious activities. In [2], a privacy protected video surveillance system is provided using camera network.

In *smart home* applications, camera network provides a system that can constantly monitor the elderly or people with special needs, and then automatically notify the hospital in case of an emergency. Camera network is also used to enhance the *tele-conferencing* experience by providing multiple views of participants, as well as improving segmentation results from cameras with multiple modality [3]. In a camera network with depth cameras, a virtual environment can be generated and updated in real-time, which provides unique opportunities to improve personal experience in various augmented reality applications.

1.1.2 Camera Network Research

The special characteristics of cameras in camera networks brings advantages as well as challenges, when compared with traditional sensor networks. Some of the most challenging issues are highligted below:

1. The light-of-sight property of cameras makes it possible to get information in distant so that camera networks can cover a much broader environment. However, due to the line of sight property of cameras, the visibility model involves the restrictions of viewing angles, projective distortion, self and mutual occlusions and specific application requirements, which demands much more sophisticated mathematical tools to model the coverage. Furthermore, it is a much more difficult problem to optimize the coverage of camera networks not only because of the complex visibility model, but also because of the huge dimension of the configuration space consisted of number of cameras, camera types, positions, yaw and pitch angles and so on.

2. Due the large amount of data collected by cameras, the data flow in the network is extremely heavy. In most cases, distributed algorithms have to be carefully crafted to alleviate the communication burden.

3. Although a single camera is able to provide large amount of data, it is far from sufficient to meet the requirement of large scale applications. Camera network is able to provide information of the scene with better coverage and different viewing perspectives. Furthermore, heterogeneous camera networks, which con-

sist of cameras of different types, offers information with different modalities. On the other hand, all those information available poses an important data fusion problem: how to make intelligent inferences by combining the data from multiple cameras in a timely manner.

To deal with these challengers, most camera network research focuses on one of two main areas: *camera planning* and *sensor fusion*.

Camera planning is the first aspect of camera network research which has enormous impact on performance. When only static cameras are considered, the problem is usually referred to a *camera placement problem*. Camera planning can be very challenging because camera networks are typically deployed in urban or indoor environments characterized by complicated topologies, stringent placement constraints and the constant flux of human or vehicular traffic. Besides, there are a myriad of camera sensors and many of them have overlapping capabilities.

Given a fixed budget with limited power and network connectivity, the choice and placement of sensors becomes critical to the continuous operation of a visual sensor network. In addition, the performance of the network depends highly on the nature of the specific tasks of the application. For example, biometric and object recognition require the objects to be captured in a specific pose; triangulation requires visibility of the same object via multiple sensors; object tracking can tolerate certain degree of occlusion using a probabilistic tracker. Lastly, many researches successfully formulate camera placement as combinatorial optimization problems, those formulations are usually NP complete with huge dimensions of solution space, which makes them

difficult to solve in reasonable time.

When the camera network involves active cameras, such as pan-tilt-zoom cameras or cameras deployed on robotic cars, techniques and theories from system control, game theory and machine learning are very helpful. Real-time decisions need to be made based on the current and previous status of the network. Due to the complexity and strict time constraint of active camera networks, it is usually impossible to find a global optimal solution and efficient approximate solutions are highly desirable.

The other area of camera network research is *Sensor fusion*, which is performed by gathering information from a number of cameras in the camera network. Traditional research in sensor network provides myriad useful generic fusion algorithms, including Kalman filter [4], Bayesian Network [5] and Dempster-Shafer theory [6]. However, the communication constraint makes it necessary to design a better distribution of the computations in the network. The fusion algorithms in camera networks need to consider not only how the information is fused in the network, but also what kind of local processing is needed to reduce the network burden. In addition to generic data reduction techniques, such as image and video compression, modern camera networks usually consist of programmable "smart cameras", which can perform application specific operations that store only useful data.

Besides, due to the projective distortion when projecting objects onto a camera's image plane, the geometric relationship between images from different camera views is very complicated. This poses new challenges concerning sensor fusion in camera networks compared with other types of sensor network. Since there is no fixed mapping between images from two camera views, efficient on-line registration algorithms

need to be carefully designed, even with complete camera calibration data. We refer to the data alignment problem in camera networks as *geometric fusion*.

1.2 Contributions of Dissertation

The research work presented in this dissertation addresses the challenges on both camera planning and sensor fusion.

In order to deal with the multiple design challenges in camera placement, a unified flexible sensor-planning framework was proposed that can incorporate all the modeling details, including to the complexity of the environment, self and mutual occlusion of moving objects, diverse sensor properties and a myriad of performance metrics for different applications. The proposed visibility model supports arbitrary-shaped 3D environments and incorporates realistic camera models, occupant traffic models, self occlusion and mutual occlusion. Using this visibility model, most static camera placement problems can be adapted under the framework of Binary Integer Programming. The proposed framework provides a complete formulation of the camera placement problem, and it covers not only the 3D static environment topology and different types of requirements, but also captures the traits of the scene dynamic, including mutual occlusion and pedestrian traffic pattens.

At the same time, although the proposed framework is general enough for most camera placement problem modeling, its complexity increases exponentially with problem size. My research is the first to study the entire spectrum of approximation algorithms from Greedy, Markov Chain Monte Carlo (MCMC) and various relaxation techniques. Our key contribution is to provide not only a generic formulation

of the camera placement problem, but also to present detailed approaches to adapt the formulation to various approximation schemes. Our adaptations to Simulated Annealing (SA) and Semi-Definite Program (SDP) are novel. We demonstrate that Greedy approach and its variants can obtain a good first order estimation. MCMC approaches are more complex but still return good solutions even in complex problems. Linear Programming (LP) and (SDP) relaxations are most complex, but they can provide good performance bounds.

Besides the general contribution to the camera placement problem, this research also makes contributions to the sensor fusion problem in camera networks. The major contribution is focused on solving the geometric fusion problem by utilizing the characteristics of camera projection to align image data from different cameras. Specifically,

1. After providing a hierarchy of classic camera models with different levels of complexity, this research proposes a novel approximation camera model, *blob homography*, which is specifically designed for geometric fusion in a typical video surveillance environment.

2. Using the blob homography model, a fusing algorithm for thermal and regular cameras is proposed for robust segmentation of humans in video sequences. By decomposing the homography matrix into rectified domain, It significantly reduces the complexity of parameter estimation, and the registration of multi-modality information becomes more robust with the help of temporal inference by including the homography parameters in a two-tier tracking system.

3. A distributed camera fusing algorithm is presented for combining the information from multiple regular cameras and depth cameras in a virtual mirror system. This dissertation shows how calibration and distributed image processing algorithms can be used in camera networks to tackle complex 3D scene acquisition and generation tasks with great efficiency.

1.3 Dissertation Overview

This dissertation is organized as follows: Chapter 2 reviews the related work in camera network research, with a focus on the camera model, camera placement and sensor fusion between thermal-RGB camera pairs and depth-RGB camera pairs. In Chapter 3, a unified framework is presented for modeling all kinds of camera placement problems using binary integer programming. Chapter 4 continues the study on the camera placement problem with a focus on the approximation techniques in solving those problems. The sensor fusion problem in camera networks is discussed in Chapter 5 and Chapter 6. In Chapter 5, a generic framework for camera fusion algorithm design is proposed, with a focus on the geometric fusion algorithms. The effectiveness of the framework is validated through practical fusion problems between visible light cameras with thermal and depth cameras in Chapter 6. The dissertation is concluded in Chapter 7

Chapter 2 Related Work

2.1 Related Work on Camera Models

Cameras are a variety of sensors which form images by collecting light rays from surroundings. Video cameras are the most common type of cameras, which work by collecting the visible light so that the generated images are similar to those from our human vision. Using special sensors to collect infrared light emissions, thermal cameras are capable of getting images with intensity value proportional to surface temperature of objects in the scene. Depth cameras estimate depth value at each pixel by collecting the reflected rays of EM radiation that they projected onto the scene. Cameras can also be found in medical applications such as gamma cameras (scintillation cameras) and X-ray cameras.

Despite the enormous differences of cameras, most cameras share the following properties,

1. Unlike other type of sensors which usually obtain signals from proximity, cameras can collect light rays emitted from objects far away and are therefore particularly useful for remote sensoring.

2. The process of collecting light rays is naturally parallel, which gives cameras the opportunity of gathering large amount of data in a very short time. The amount of information collected by cameras is usually several magnitudes larger than other type of sensors.

3. Due to the line of sight property of cameras, the visibility of objects to cameras are usually affected by not only the distance but also viewing angles and occlusions.

Although cameras may use EM radiation with different frequencies, their imaging model are astonishingly similar. Most of the models mentioned in this section comes from the study of visible-light camera but are equally applicable for other types of cameras.

Pinhole camera model [7, ch.6], also known as finite projective camera model, is the most popular model for cameras. The model is believed to be introduced by Mozi (400 BC) in China and Aristotle (200 BC) in Greece independently. In fact, the word *camera* comes from this model as *camera obscura*, which means "darkened chamber". It is a description of an imaging device of a light-proof box with a small pinhole on one side. Light rays from surrounding scene pass through a single point and project an inverted image on the opposite side of the box, as in Figure 2.1

Although modern cameras are much more complicated than pinhole camera, the model provides very good approximations for most cameras with tremendous mathematical conveniences, thanks to the work in [8, 9, 10, 7]. It can be mathematically formulated as a 3×4 projection matrix with rank 3 and 11 degrees of freedom in homogeneous coordinate. The detailed form will be explained in Chapter 3.

Starting from the pinhole camera model, research took two separate paths to advance the model. One research direction focuses on the simplification of the pinhole model in order to expedite the calculation for real time applications. For instance,

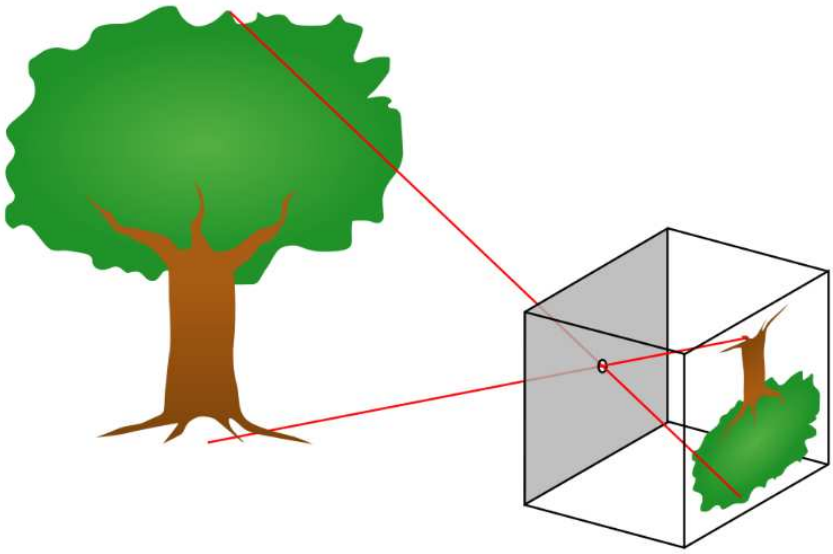


Figure 2.1: Pinhole camera model

by moving the center of projection to infinity, incident rays from the scene become parallel. Although no cameras actually follow this model, it is a reasonably close approximation when the object is far away from the camera and the complexity of the model can be dramatically reduced. Under this assumption, *orthographic projection* [10] uses only five degrees of freedom which incorporates translation and rotation. A more realistic model is the *weak perspective projection* [10, ch.1] model with only seven degrees of freedom. It is an extremely useful model when the scene depth variation is small relative to the average depth from the camera. Similar approximation can be found in *para-perspective* [11] with a better model of the perspective distortion. All of those coarser models can be generalized in *Affine camera model* [12], which has eight degrees of freedom. Mathematically, the difference between affine model and finite projective model is that the left handside 3×3 submatrix of the projection matrix is non-singular in finite projective camera model and singular in affine camera

models.

The other direction of camera model research is to exploit more sophistication to cameras in real life. First of all, most modern cameras are equipped with lens, which can keep the picture in sharp focus while gathering light from larger area. However, the introduction of lens makes the pinhole camera model somewhat insufficient. Even under the most simplified thin lens model [10, ch.1.2.2], only objects within a certain depth range will remain sharp, while objects out of the range will be blurred. Also, lens distortion sometimes can be significant so that the camera no longer preserves linearity. A lot of models are introduced to formulate this effect and make reasonable corrections [13, 14], which bring more complexity to the camera model. Also, some researches exploit the cameras with non-planar image planes. Spherical projection [10, ch.1.1.3] model, for instance, uses spherical image sensor and gives some advantages on preserving the size of object in the generated image. However, it does not preserve straight lines. In [15], Ponce gives an excellent survey and generalization to a range of more complex camera models in theory and real life.

2.2 Related Work on Camera Planning

The research of camera planning focuses on placing and controlling the cameras in a camera network. For a network only consists of static cameras, only camera placement is needed. However, when active cameras such as pan-tilt-zoom cameras or robotic cameras are concerned, camera planning involves on line algorithms which change the parameters of those active cameras accordingly. Since my work mostly focuses on the static camera planning, in this section, I present a detailed survey in the camera

placement, together with a brief review of techniques used in active sensor planning.

2.2.1 Related Work on Camera Placement

The problem of finding the optimal camera placement has been studied for a long time. The earliest investigation can be traced back to the "art gallery problem" in computational geometry. This problem is the theoretical study on how to place cameras in an arbitrary-shaped polygon so as to cover the entire area [16, 17, 18]. It covers a set of important topics in computational geometry including Delaunay triangulation, vertex covering and so on. Although Chvátal has shown in [19] that the upper bound of the number of cameras is $\lfloor n/3 \rfloor$, determining the minimum number of cameras turns out to be a NPcomplete problem [20]. While the theoretical difficulties of the camera placement problem are well understood, few solutions can be directly applied to realistic computer vision problems since the original formulation of the "art gallery" problem lacks for realistic models for either the cameras or the environment under surveillance, and provides few efficient computational approaches to calculate optimal placements under different scenarios.

Camera placement has also been studied in the field of photogrammetry in order to obtain the most accurate 3-D reconstruction of the scene. Various metrics such as visual hull [21] and viewpoint entropy [22] have been developed and optimization are realized by various types of ad-hoc searching and heuristics [23]. These techniques assume very dense placement of cameras and are not applicable to wide-area wide-baseline camera networks.

Recent widespread deployments of video camera networks, however, turn camera

placement from a problem of theoretical interest into an important tool that can significantly improve the performance, coverage and cost effectiveness of the network.

Recently, Ramakrishnan et al. propose a framework to study the performance of sensor coverage in wide-area sensor networks [24]. Unlike previous techniques, their approach takes into account the orientation of the object. They develop a metric to compute the probability of observing an object of random orientation from one sensor, and use that to recursively compute the performance for multiple sensors. While their approach can be used to study the performance of a fixed number of cameras, it is not obvious on how to extend their scheme to find the optimal number of cameras as well as how to incorporate other constraints such as the visibility from more than one camera.

More sophisticated modeling pertinent to visual sensor networks are recently proposed in [25, 26, 27]. The sophistication in their visibility models comes at a high computational cost for the optimization.

While the original "art gallery" problem was formulated in the continuous 2-D or 3-D spaces, most recent approaches consider the problem entirely in discrete domain – instead of optimizing continuous functionals using calculus of variation, discrete-domain approaches quantize the search space into finitely many "candidate" positions and search for the best configurations that optimize a given cost function. This strategy naturally leads to combinatorial problems with the camera, environment, and traffic models encoded in different integral constraints and objective functions. Efforts have been made to formulate the discrete camera placement problems using standard binary linear programming [28, 29, 30] and quadratic programming [27].

For the few camera placement formulations in the continuous space, traditional numerical methods are used taking advantage of the derivative of those continuous functions. For example, the simulated quenching scheme used in [26] takes several hours to find the optimal placements of four cameras in a room. Other optimization schemes such as hill-climbing[25], semi-definite programming[27] and evolutionary approach[31] all prove to be computationally intensive and prone to local minima.

The majority of camera placement researches adopt formulations in discrete domain. While most formulations result in NP-hard problems, a myriad of practical solutions including Binary Integer Programming solvers (BIP), greedy approach, greedy heuristics, Monte Carlo (MC) simulations and Semi-Definite Programming relaxations (SDP) have been proposed [28, 32, 33, 34, 35, 30, 36, 27]. A summarize of camera placement optimization solution is shown in Table 2.1. Each method has its own merits in terms of ease of formulation, computational complexity, worst or average case performances, scalability, etc. To further complicate matters, different researchers often tackle slightly different objective functions and design specific approximation techniques accordingly. To the majority of the vision community, it is difficult to discern the merits of different approaches and to identify the appropriate solution for a specific placement problem at hand.

2.2.2 Related Work on Active Camera Planning

The topic of controlling active cameras to conduct active vision has been studied for decades. Due the real time constraint and much bigger state space to explore, most techniques avoid the full scale optimization methods we mentioned in last section

Table 2.1: Various approaches for solving the camera placement problem

	BIP/LP	Greedy	Heuristic	MC	SDP
[28]	X				
[32]	X				
[33]	X		X		
[34]	X	X			
[35]	X	X			
[30]	X				
[36]		X		X	
[27]		X	X		X

but instead use greedy or heuristic method to control the cameras without strict mathematical proof of optimality [37, 38, 39]. The most popular theoretical scheme fit for active vision problem is Markov Decision Process based reinforcement learning [40]. In this formulation, there are a set of action space and state space, and the transition of the states not only depends on the previous state but also the current action. There are also different real valued reward for a specific action-state pair(reinforcement). The goal of this process is to decide a sequence of actions in order to achieve the maximum expected accumulated rewards. The most aggressive formulation of reinforcement learning is to obtain both the suboptimal strategy and the state's dynamics at the same time, with no prior knowledge of the state transition dynamics. The only assumption is the dynamic transition is a time-invariant Markov chain. Classic solutions for this problem includes Temporal-difference learning, Q-learning and their variations. The convergence of these methods have been proved both theoretically and experimentally [41] but their efficiency in real applications are frequently challenged in terms with both its sensitivity to space size and convergence time [40]. Foresti et at. [42]implement a Q-learning algorithm to control one pan-

tilt-zoom for pedestrian foveation. In their implementation, they use a training stage to achieve better estimation of the state transition before deciding the appropriate control of the PTZ cameras. Although their pioneer work brings some insight into the problem, their work only deal with one PTZ camera and the action speed of PTZ camera and its power assumption are not considered when modeling the action space as they simply assume the camera can foveate in any position in any time instance. Naish [43] proposed a Neuro-dynamic [44] reinforcement control strategy for range sensors. They had limit action space and experimental results are provided by only synthesis data.

2.3 Related Work on Sensor Fusion in Camera Network

2.3.1 General Sensor Fusion

The problem of sensor fusion has been studied as a general problem for decades. Since sensor fusion heavily depends on the application and type of sensors, most of those studies remain in a very high level. On one hand, the conclusions from sensor fusion study are general enough to cover all kinds of sensor networks including camera network. However, on the other hand, those models are usually very abstract and have little guidance for a specific sensor network design. Here we present a brief survey of some of the most well-known models is presented. Interested readers can refer to [45, 46] for detailed treatment.

The sensor fusion model proposed by US Joint Directions of Laboratories is arguably the most classic sensor fusion model, better known as the *JDL model* [47]. It

was later revised in [48, 49]. The JDL model is consisted of five different levels [45],

1. Source preprocessing *(Level 0)*: The pre-pocessing step which is performed locally at each sensor. Common process includes signal denoising, compression and so on. The objective of this process is to reduce the communication and computation burden in the later fusion phase.

2. Object/Entity Assessment *(Level 1)*: A process includes data alignment, data association, object tracking and identification. This process accounts for most of the research topics in academia.

3. Situation Assessment*(Level 2)*: In this level, an attempt is made to find a contextual description of the relationship between the objects and observed events.

4. Threat Assessment *(Level 3)* : By combining the priori knowledge and predictions about the future situation, this step infers about vulnerabilities and opportunities for operation in the sensor network. Typical examples include estimation of security thread level and locating the target.

5. Process Refinement *(Level 4)*: It is a process focus on the fusion process itself instead of the data from the sensor. Sensor management is the central functionality in this level, which will is the most task in an active sensor network.

The JDL model provides a comprehensive infrastructure for all sensor fusion applications. However, it is so generic that the methodology various dramatically even

within the same level. Some of the typical sensor fusion techniques in camera network corresponding to each level in JDL model are illustrated in Figure 2.2

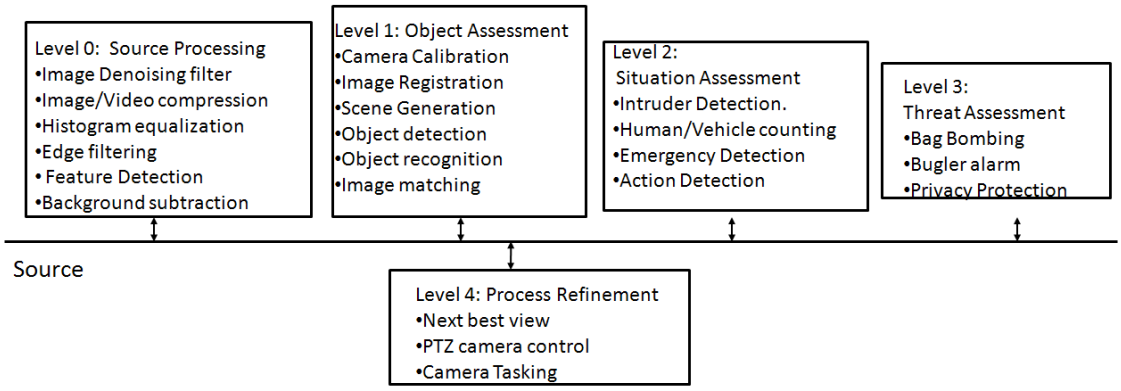


Figure 2.2: Sensor fusion techniques in camera network

In the following sections, we will survey some of the related camera network fusion techniques closely related to the application project in this dissertation. Namely, the camera image fusion between thermal cameras, visible light cameras and depth cameras.

2.3.2 Fusion of Thermal Camera and Regular Camera

For decades, the problem of segmenting human in video sequence has been a central issue in computer vision. Despite its popularity, it remains to be a challenging problem because visual appearances are subjected to occlusion, illumination change, hightlight, shadow and color confusion. Recently, systems using sensors of different modalities have been proposed to improve human video segmentation results. Among them, thermal infrared sensors are particularly popular as human bodies usually present different temperature characteristics from that of the environment.

The introduction of infrared camera provides both opportunities and challenges. On one hand, the extra modality provided by infrared camera offers supplementary information about the human body and thus should potentially improve the classification. On the other hand, the information from the visible-light and thermal cameras are not spatially aligned and the new modality can bring new channels of noises which could further confuse the classifier. In this paper, we tackle the registration problem by learning blob-to-blob homographies according to the disparity of each blob to attain a pixel level registration. The multi-modality information is then combined under a two tier tracking algorithm and an unified background model to mitigate segmentation noise from either modalities.

Most existing systems solve the registration problem by either optical fusion[50, 51] or image warping[52, 53, 54]. The optical fusion methods use specially-designed optical device to merge the optical axes of the two cameras so that the two cameras can see exactly the same view. Despite its computationally efficiency and registration accuracy, it suffers from high manufacture costs. The image warping method calculates a homography matrix during the calibration procedure by point matching. The homography is used to warp segmentation results from one modality to the other. The same homography matrix is applied to all objects in the scene regardless of their depths. Due to over-simplification from 3D projection to 2D mapping, the performance dwindles when there exists significant variation of objects' depth in the sequences. Some systems such as [54] and [52] adopt additional search procedures to correct the registration error. Those searching algorithm significantly diminish the algorithmic efficiency and will fail when either of two views has defective foreground

segmentation.

In [55], instead of performing the warping in image scale, the authors align the two foreground blobs by identifying shape feature points to estimate the homography, through either skeleton or discrete curve evolution. Although our algorithm follows a similar idea, we further exploit the camera model and reduce the number of parameters need to be learned from eight to just one. Furthermore, by including the parameter into a tracker, we make full use of the temporal information to infer the homography so that the registration would still work in noisy frames where no valid observation is available.

Traditional sensor fusion techniques are pervasively used to improve the segmentation from information obtained by multiple sensor. Kumar et al. [53] adopt fuzzy logic to evaluate the confidence from each sensor. Han and Bhanu [54] compares different rules under Bayesian framework, while combined trackers such as Kalman filter and Particle filter are used to fuse the multi-modality observations in [56, 57, 58]. Alternatively, the fusion can be performed from the image perspective. In [50], image segmentation is performed using the output of thermal camera as seeds. Morphological operations are adopted in [52] and [59].

2.3.3 Fusion of Depth Camera and Regular Camera

The introduce of Time-of-flight sensor has brought a lot of enthusiasm to the camera network community. Time-of-flight(ToF) sensors provide independent range estimates at each pixel in real time, which provides critical complementary depth information for regular cameras in camera network.

Early studies on sensor fusion between ToF sensor and regular camera focus on the accuracy of depth estimation [60, 61, 62, 63, 64]. In [60], data alignment is achieved by estimating disparity from the TOF-depth measurements. Then the depth information is converted into disparity and used for improving the stereo matching process. On the other hand, Zhu et. al [63] convert the disparity information from stereo camera into depth and directly fuse with the depth value in ToF sensor under a probabilistic Markov Random Filed.

As the ToF and other depth sensors become more reliable and available, researchers begin to look at higher level applications. For instance, depth camera has significantly improved the foreground/background segmentation in video processing. In [65], an initial foreground probability is obtained by looking at the depth data, then the depth probability is smoothed by considering the color consistency in neighborhood pixels. An edge preserving filter is then used to further smooth the segmentation result while preserve the boundary. Instead of directly applying smoothing filter, Wang et. al [66] uses an energy minimization framework to incorporate the depth, color information and spatial smoothness, the final formulation is solved by graph cut [67]. Similar system is seen in [68].

The introduction of Microsoft Kinect [69] dramatically reduces the cost of deploying depth camera in camera network. Since its first apprance in November, 2010, academic researchers, professionals and amateurs have been pouring ideals into this fascinate field and a lot of interesting applications have emerged on the Internet. For example, Using Kinect to improve the gesture recognition accuracy, a lot of novel applications can be crafted as novel human-computer interaction applications, such

as virtual piano [70], robot control [71] and hand controlled browser [72]. Other applications utilize the capability of depth camera equipped camera network to quickly obtain the 3D information of the environment, such as 3D visual "Simultaneous Localization and Mapping" [73], 3D video conferencing [74] and the virtual mirror we proposed in Section 6.6.2 .

Several papers have demonstrated the concept of virtual mirror for various applications. Though they differ in some aspects, most of them only deal with simple appearance modification with a limited viewpoint [75, 76, 77]. Darrell et al. [75] described a virtual mirror interface that reacted to people by applying different graphical effect on their faces. Similarly, in [76], the authors proposed a virtual facial modification program by user-driven 3D-aware 2D warping. However, both of them did not consider the view point's influence on rendering virtual mirror. Francois and Kang [78] designed a handheld mirror simulation device. Although it considered the viewpoint change during the mirror image transformation, the system simply model the world as a plane parallel to the mirror/imaging.

Chapter 3 Formulating the Camera Placement Problem

3.1 Introduction

In recent years we have seen widespread deployment of camera networks for a variety of applications. Proper placement of cameras in such a distributed environment is an important design problem. Not only does it determine the coverage of the surveillance, it also has a direct impact on the appearance of objects in the cameras which dictates the performance of all subsequent computer vision tasks.

However, even with decades of study on camera network, the most ambitious goal of designing a universal camera network configuration tool has never been achieved. The intricacy comes from a lot of factors. Firstly, unlike other sensors, the line-of-sight property of cameras makes them much more vulnerable to occlusions by both static and dynamic objects. This is especially tricky when considering camera placement in wide area indoor or outdoor environment characterized by complicated topologies, stringent placement constraint, and a constant flux of occupant or vehicular traffic. Secondly, camera is a abundant category ranging from infrared to range sensing, from static to pan-tilt-zoom, from telescope to omni-directional. There is a lack of a generic model to abstract their overlapping capability and differentiate their own characteristic at the same time. More importantly, the performance of the network depends heavily on the nature of the specific tasks in the application. The term "visible" has dramatically different meaning in different settings. For example, in

facial recognition application, one will require getting a clear frontal shot of the person while in gait analysis, the profile view becomes the objective.

In this chapter, we tackle this problem by presenting a generic four-step scheme consisted of

1. A generic visibility modeling.

2. A discretization process.

3. A Binary Integer Programming (BIP) formulation.

4. An optimization tool to solve the problem efficiently.

Instead of trying to solve all the problems at once, we argue that most of the camera planning problems can be decomposed into the four steps and solved efficiently.

3.2 General Visibility Model

The first question we consider in planning a camera network is *"when will an object be visible to a camera?"* It is generally not a very difficult problem given some knowledge of the sensor but still takes some effort to describe it in a mathematical fashion.

We define our general visibility model to be a function which takes a fixed camera and target with known parameters as input and outputs a continuous or binary value to indicate the visibility of the target from specific sensor position. Our visibility model does not only deal with different cameras from infrared to range sensing, from static to pan-tilt-zoom, but also incorporates the perspective requirement from dif-

ferent vision tasks such as profile view for gait analysis and frontal view for facial recognition.

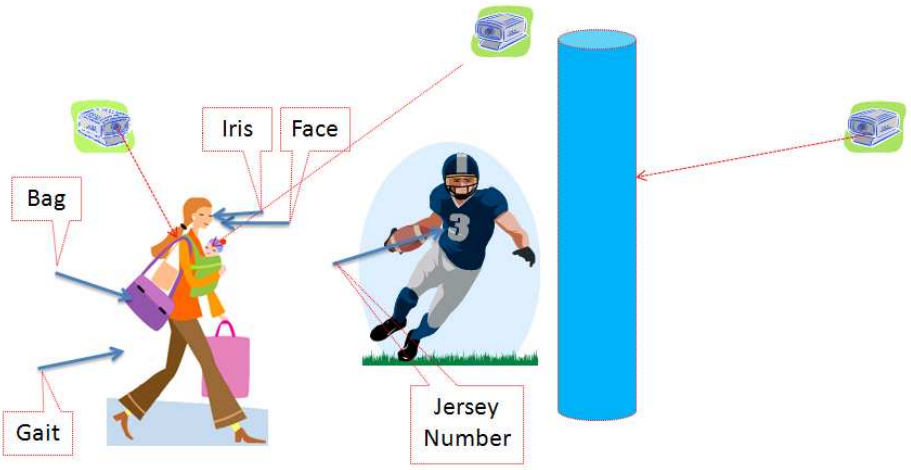


Figure 3.1: Three-dimensional visibility model. *The target object can be anything depending on the application, such as bags, iris faces jersey numbers and so on. Cameras of arbitrary yaw and pitch angles can be placed anywhere in the 3-D environment.*

Consider the 3D environment depicted in Figure 3.1, the first thing our visibility model requires is to identify the target object of this specific surveillance task. For instance, instead of the entire body, the system may only care for particular biometric features such as face or eye for biometric application. In anti-terrorist application, the target object should be the bag the individual is carrying. The sports broadcasting system may want to track the jersey number of the player.

After identifying the target object for observation, several factors contribute to the visibility of the object. They are usually the input of a camera configuration system.

1. Camera geometry. It includes the camera's 3D position and pose. We can usually describe it by a 3D coordinate of the camera's center and a 3D vector

called "direction of projection", which is the norm of camera's image plane.

2. Target geometry. Similar to the camera geometry, the target geometry consists of a 3D position of the target's center and a vector indicating the object's orientation.

3. Field of view (FOV). It is a volume within which objects can be seen from a particular camera. The FOV is determined by the camera's focal length and the size of its image plane. It's usually denoted using two angles which are the horizontal and vertical angle just as in Figure 3.2.

4. Angle of view (AOV). Similar to FOV, angle of view is an angular extension of a visible volume from which the object can be seen, as in Figure 3.2. It is a function of shape of the target object. An planar object usually has 180 degree of AOV; a concave surface usually has less than 180 AOV and a convex object has larger than 180 degree; a sphere has 360 degree of visible angle which means it can be visible from any observation angle.

5. Environmental topology. It is usually a floor map containing all the obstacles. For computational convenience it is approximated by combination of primer shapes such as polyhedral, spheres and planes. Sometimes the environmental topology can also include dynamic obstacles such as other pedestrian in the environment.

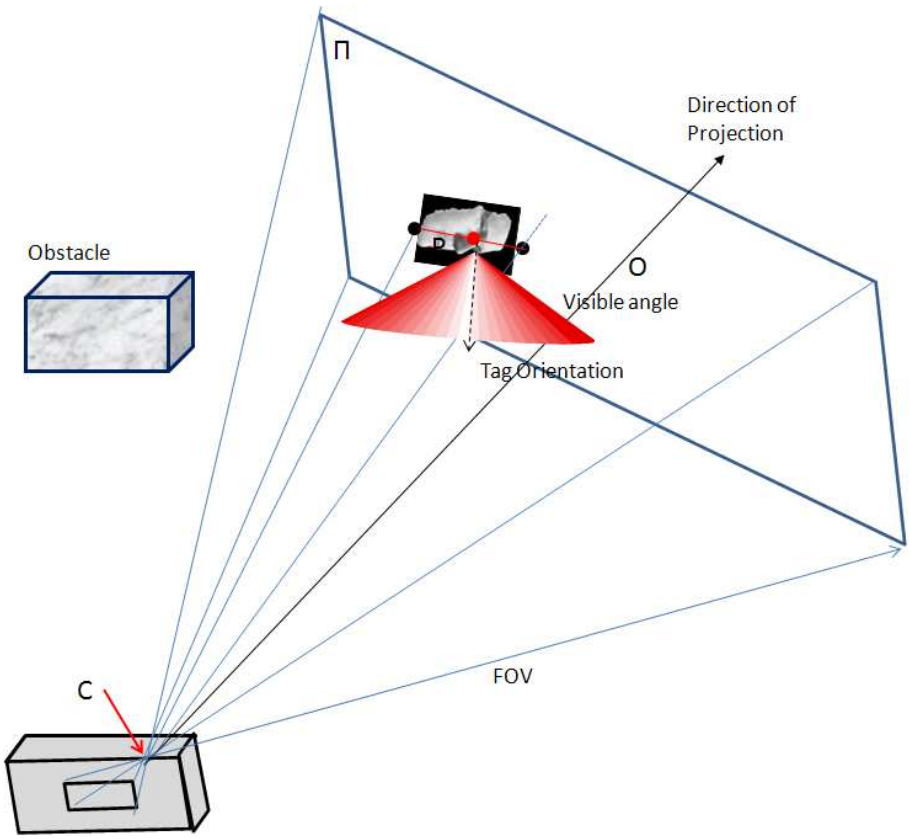


Figure 3.2: FOV and AOV of camera. *FOV is the pyramid depicted by blue lines and the angle of view of the object is shown in the red cone. the red cone from the object is the AOV*

3.2.1 Visibility Model for a Single Camera

In this section, we outline a general model to compute performance of a given camera in a confined three-dimensional environment.

Table 3.1 provides a quick summary of all the symbols used in our derivation.

We assume that the 3-D environment has vertical walls with piecewise linear contours. Obstacles are modeled as columns of finite height and polyhedral cross sections. Whether the actual target is the face of a subject or an artificial object, it is reasonable to model each target as a small flat surface perpendicular to the ground plane. We further assume that all the targets are of the same square shape with

Table 3.1: Symbols for camera visibility

P	target center
$\mathbf{v_P}$	target pose vector with $\|\|\mathbf{v_P}\|\| = 1$
Γ	Horizontal Plane where P lies
β	Occlusion angle measured at P on Γ
β_s	Starting position of the occlusion angle
$\mathbf{v_V}$	Normal of Γ with $\|\|\mathbf{v_V}\|\| = 1$
l	Line segment at the intersection between the target and Γ
P_{l1}, P_{l2}	Two end points of l
C	Camera's Center of Projection
$\mathbf{v_C}$	Camera's direction of the projection or pose vector with $\|\|\mathbf{v_C}\|\| = 1$
α	Angle between $\mathbf{v_P}$ and the vector from P to C
Π	Image plane, a finite-size rectangle with normal $\mathbf{v_C}$ and its distance from the center of projection defines the focal length.
O	Center of the Π plane
P', P'_{l1}, P'_{l2}, l'	Projection of P, P_{l1}, P_{l2} and l on Π
K	Fixed environment parameters about the walls and obstacles
Υ	A point in the camera space parameterized by C and $\mathbf{v_C}$, along with other derived quantities like Π and O
Λ	A point in the target space parameterized by P, $\mathbf{v_P}$ and β_s
Θ	Angle of view for the target object.

known edge length $2w$. Without any specific knowledge of the height of individuals, we assume that the centers of all the targets lie on the same plane Γ parallel to the ground plane. This assumption does not hold in real world as individuals are of different height. Nevertheless, as we will demonstrate in Section 3.5.1, such height variation does not much affect the overall visibility measurements. While our model restricts the targets to be on the same plane, we place no restriction on the 3-D

positions, yaw and pitch angles of the cameras in the camera network.

Given the number of cameras and their placement in the environment, we define the visibility V of a target using *an aggregate measure of the projected size of a target on the image planes of different cameras*. The projected size of the target is very important as the image of the target has to be large enough to be automatically identified at each camera view. Due to the camera projection of the 3-D world to the image plane, the image of the square target can be an arbitrary quadrilateral. While it is possible to precisely calculate the area of this image, it is sufficient to use an approximation for our visibility calculation. Thus, we measure the projected length of the line segment l at the intersection between the target and the horizontal plane Γ. The actual 3-D length of l is $2w$, and since the center of the target always lie on l, the projected length of l is representative of the overall projected size of the target.

Next we identify the set of random and fixed parameters that affects V. The fact that we have chosen to measure the projected length of l instead of the projected area of the target greatly simplifies the parametrization of V. Given a camera network, the visibility function of a target can be parameterized as $V(P, \mathbf{v_P}, \beta_s | w, K)$ where P, $\mathbf{v_P}$, β_s are random parameters about the target; K and w are fixed environmental parameters. These parameters are defined in the sequel and illustrated in Figure 3.1.

P defines the 2D coordinates of the center of the target on the plane Γ. $\mathbf{v_P}$ is the pose vector of the target. As we assume the target is perpendicular to the ground plane, the pose vector $\mathbf{v_P}$ lies on the plane Γ and has a single degree of freedom – the orientation angle θ with respect to a reference direction. Note the dependency of V on $\mathbf{v_P}$ allows us to model self-occlusion – the target is being occluded by the

person who is wearing it. The target will not be visible to a camera if the pose vector is pointing away from the camera. While self occlusion can be succinctly captured by a single pose vector, the modeling of mutual occlusion involves the number of neighboring objects, their distances to the target and the positions of the cameras. The precise modeling of mutual occlusion can be extremely complicated. In our model, we choose the worst-case approach by considering a fixed occlusion angle β measured at the center of the target on the Γ plane. Mutual occlusion is said to occur if the projection of the line of sight on the Γ plane falls within the range of the occlusion angle. In other words, we model the occluder as a cylindrical wall of infinite height around the target partially blocking a fixed visibility angle of β at random starting position β_s. w is half of the edge length of the target which is a known parameter. The shape of the environment is encapsulated in the fixed parameter set K which contains a list of oriented vertical planes that describe the boundary wall and obstacles of finite height. It is straightforward to use K to compute whether there is a direct line of sight between an arbitrary point in the environment and a camera.

To correctly identify and track any target, a typical classification algorithm would require the target size on the image to be larger than a certain minimum size, though a larger projected size usually does not make much difference. For example, a color target detector needs a threshold to differentiate the target from noises, and a face detector needs a face image large enough to observe the facial features. On the other hand, the information gain does not increase as the projected object size increases beyond a certain value. Therefore, the threshold version can represent our problem

much better than the absolute image size. Assuming that this minimum threshold on image size is T pixels, this requirement can be modeled by binarizing the visibility function as follows:

$$V_b(P, \mathbf{v_P}, \beta_s | w, K, T) = \begin{cases} 1 & \text{if } V(P, \mathbf{v_P}, \beta_s | w, K) > T \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Finally, we define η, the *mean visibility*, to be the metric for measuring the average visibility of P over the entire parameter space:

$$\eta = \int V_b(P, \mathbf{v_P}, \beta_s | w, K, T) \cdot f(P, \mathbf{v_P}, \beta_s) \, dP \, d\mathbf{v_P} \, d\beta_s \tag{3.2}$$

where $f(P, \mathbf{v_P}, \beta_s)$ is the prior distribution that can incorporate prior knowledge about the environment – for example, if an application is interested in locating faces, the likelihood of the head positions and poses are affected by furnishings and attractions such as television sets and paintings. Except for the most straightforward environment such as a single camera in a convex environment discussed in [79], Equation (3.2) does not admit a closed-form solution. Nevertheless, it can be estimated by using standard Monte-Carlo sampling and its many variants. The details of our Monte-Carlo sampling strategy is discussed in Section 3.5.

3.2.2 Visibility for a Camera Network

In order to calculate performance metric for the entire camera network defined in Equation 3.2, we need to give concrete formula for each of term. In this section, we first take a closer look at the term $V_b(\cdot)$. Given a single camera with the camera center at C, it is straightforward to see that a target at P is visible at C if and only if the following four conditions hold:

1. The target is not occluded by any obstacle or wall. (Environmental Occlusion)

2. The target is within the camera's field of view. (Field Of View)

3. The target is not occluded by the person wearing it. (Self-Occlusion or Angle of View)

4. The target is not occluded by other moving objects. (Mutual Occlusion)

Thus, we define the visibility function for one camera to be the projected length $||l'||$ on the image plane of the line segment l across the target if the above conditions are satisfied, and zero otherwise. In the sequel, we demonstrate how the projected length is calculated and show how we check each of the four conditions.

Using pinhole camera model, Figure 3.3 shows the projection of l , delimited by P_{l1} and P_{l2}, onto the image plane Π. Based on the assumptions that all the target centers has the same elevation and all target planes are vertical, we can analytically derive the formulae for P_{l1}, P_{l2} as follows: as l is perpendicular to both the unit pose vector of the target $\mathbf{v_P}$ and the unit normal vector $\mathbf{v_V}$ to the plane Γ, we have $P_{l1,2} = P \pm w(\mathbf{v_P} \times \mathbf{v_V})$. Their projections P'_{l1} and P'_{l2} lie on the intersections between the image plane Π and the light rays CP_{l1} and CP_{l2} respectively. For $i = 1, 2$, any point X on Π must satisfy $\langle \mathbf{v_C}, X - O \rangle = 0$ where $\langle \cdot, \cdot \rangle$ indicates inner product, and any point X on the line CP_{li} must satisfy $X = C + \lambda(P_{li} - C)$. Thus, P'_{li} for $i = 1, 2$ can be calculated as follows:

$$P'_{li} = C - \frac{\langle \mathbf{v_C}, O - C \rangle}{\langle \mathbf{v_C}, P_{li} - C \rangle}(P_{li} - C) \tag{3.3}$$

The projected length $||l'||$ is simply $||P'_{l1} - P'_{l2}||$.

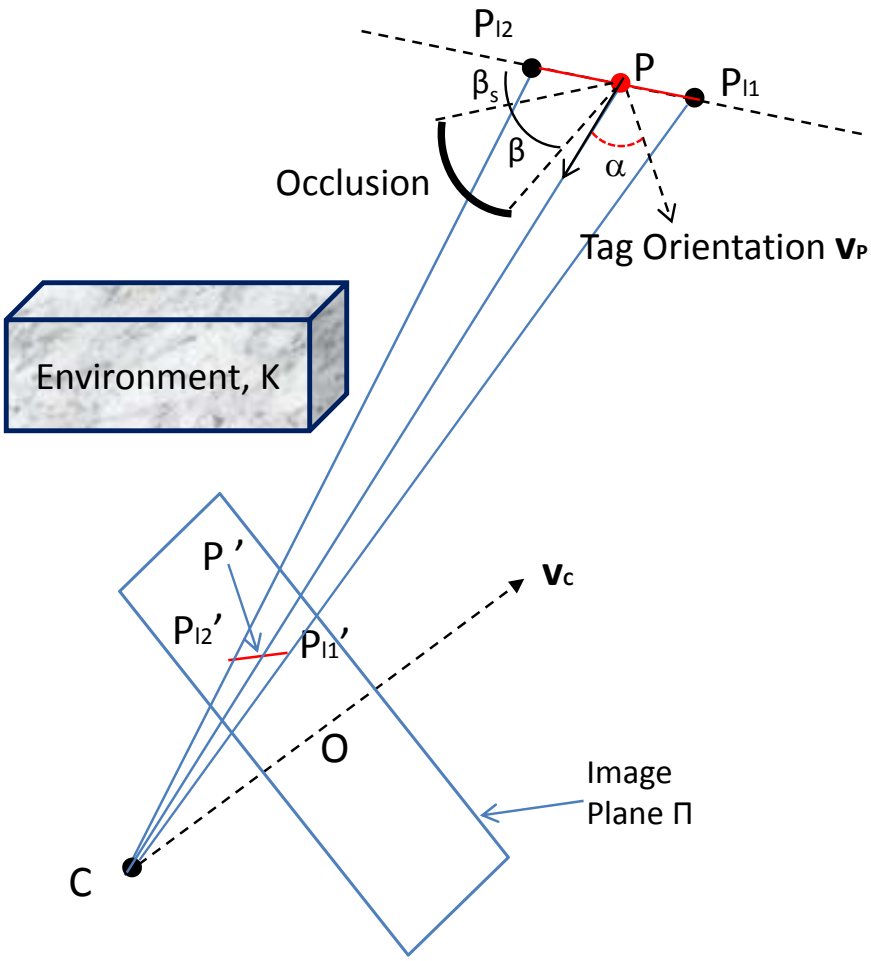


Figure 3.3: Projection of a single target onto a camera.

The projected length l under other camera model can be similarly modeled.

After computing the projected length of the target, we proceed to check the four visibility conditions as follows:

1. **Environmental Occlusion:** We assume that environmental occlusion occurs if the line segment connecting camera center C with the target center P intersect with some obstacle. While such an assumption does not take into account of partial occlusion, it is adequate for most applications where the target is much smaller than its distance from the camera. We represent this requirement as

the following binary function:

$$\text{chkObstacle}(P, C, K) = \begin{cases} 1 & \text{No obstacles intersect} \\ & \text{with line segment PC} \\ 0 & \text{else} \end{cases} \tag{3.4}$$

Specifically, the obstacles are recorded in K as a set of oriented vertical planes that describe the boundary wall and obstacles of finite height. Intersection between the line of sight PC and each element in K is computed. If there is no intersection within the confined environment or the points of intersection are higher than the height of the camera, no occlusion occurs due to the environment.

2. **Field of View:** Similar to determining environmental occlusion, we declare the target to be in the field of view if the image P' of the target center is within the finite image plane Π.

   Using a similar derivation as in (3.3), the image P' is computed under the pinhole camera model as follows:

$$P' = C - \frac{\langle \mathbf{v_C}, O - C \rangle}{\langle \mathbf{v_C}, P - C \rangle}(P - C) \tag{3.5}$$

   We then convert P' to local image coordinates to determine if P' is indeed within Π. We encapsulate this condition using the binary function $\text{chkFOV}(P, C, \mathbf{v_C}, \Pi, O)$ takes camera intrinsic parameters, target location, pose vector as input, and returns a binary value indicating whether the center of the target is within the camera's field of view.

When a camera model with lens is used, we need to also check the depth of the target, $||P - C||$, is within the range of the depth of view $[d_{min}, d_{max}]$.

3. **Angle of view or Self Occlusion:** As illustrated in Figure 3.3, the target is self occluded if the angle α between the light of sight to the camera $C - P$ and the target pose $\mathbf{v_P}$ exceeds $\frac{\pi}{2}$. In other word, we say the object is not self occluded if the camera is within the angle of view of the object. We can represent this condition as a step function $U(\frac{\Theta}{2} - |\alpha|)$.

4. **Mutual Occlusion:** In Section 3.2, we model the worst-case occlusion using an angle β. As illustrated in Figure 3.3, mutual occlusion occurs when the target center or half the line segment l is occluded. The angle β is suspended at P on the Γ plane. Thus, occlusion occurs if the projection of the light of the sight $C - P$ on the Γ plane at P falls within the range of $[\beta_s, \beta_s + \beta)$. We represent this condition using the binary function chkOcclusion$(P, C, \mathbf{v_P}, \beta_s)$ which returns one for no occlusion and zero otherwise.

Combining both $||l'||$ and the four visibility conditions, we define the projected length of an oriented target with respect to camera Υ as $I(P, \mathbf{v_P}, \beta_s | K, \Upsilon)$ follows:

$$\begin{aligned} I(P, \mathbf{v_P}, \beta_s | w, K, \Upsilon) = ||l'|| \cdot \\ \text{chkObstacle}(P, C, K) \cdot \text{chkFOV}(P, C, \mathbf{v_C}, \Pi, O) \cdot \\ U\left(\frac{\Theta}{2} - |\alpha|\right) \cdot \text{chkOcclusion}(P, C, \mathbf{v_P}, \beta_s) \end{aligned} \tag{3.6}$$

where Υ includes all camera parameters including Π, O and C. As stated in Section 3.2, a threshold version is usually more convenient:

$$I_b(P, \mathbf{v_P}, \beta_s | w, K, \Upsilon, T) = \begin{cases} 1 & \text{if } I(P, \mathbf{v_P}, \beta_s | w, K, \Upsilon) > T \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

3.3 Discretize the Space by Sampling

It is generally not difficult to derive a function to predict the visibility of a target object provided when both the camera and target parameters are given and to further decide the "observability" of the target object to a fixed camera network. However, the goal of an optimal camera placement is to identify, among all possible camera network configurations, the one that maximizes the average observability assuming the target parameters to be random with or without prior knowledge. In reality, the visibility function never possesses an analytic form because it usually involves some thresholding, enumeration and etc. Therefore it is very difficult to apply conventional continuous optimization strategies such as variational techniques or convex programming. As such, discrete approaches are more suitable by finding an approximate solution over a discretization of two spaces – the space of possible camera configurations and the space of target object's location and orientation.

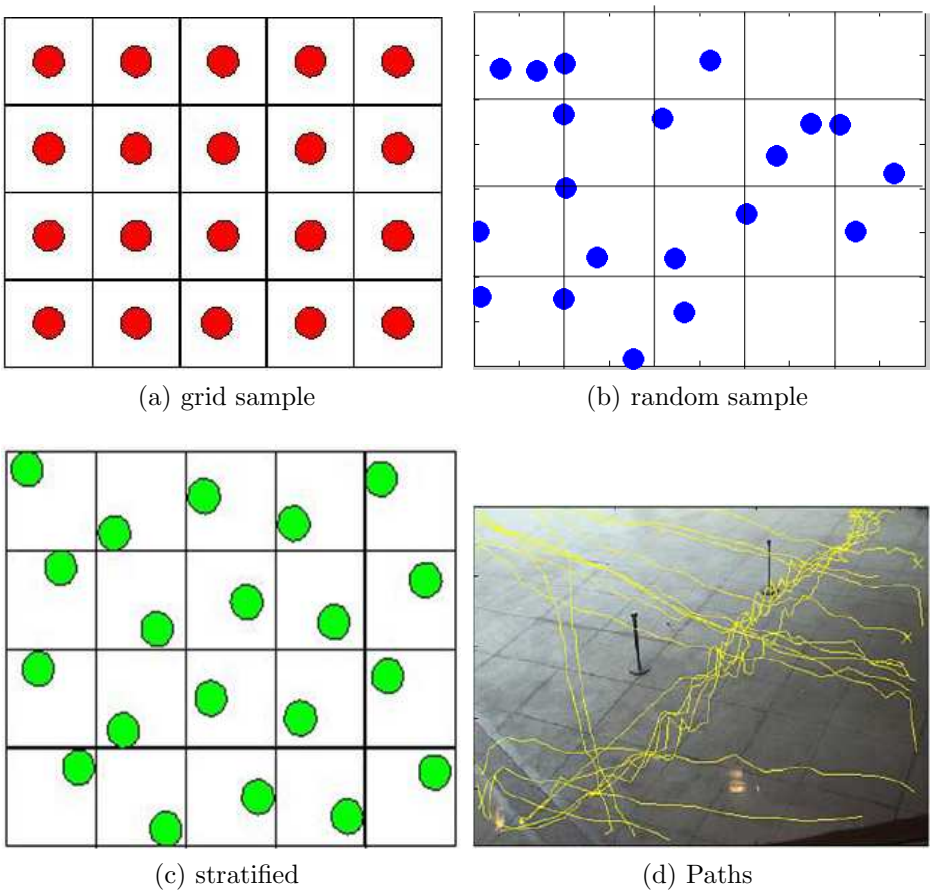
The infinite camera configuration space can be discretized into finite space with some sacrifice of accuracy. Normally, the camera configuration space — including the 3D location, pitch and yaw angle — can be uniformly sampled as candidate camera grid points. The sampling density can be defined by the user, trading off between computational complexity and approximation accuracy. For certain environmental

topology, it is easy to observe that some locations or poses may have more advantargeteous over others, such as vertices of polyhedral and the pose pointing to the center of a sphere. The discretization process should be capable to identifying these points or at least allow the user to pick these locations and poses manually.

For planning of heterogeneous camera network, the discretization can naturally model the selection of different camera types. For given fixed camera setup including position, pose and occlusion angle, we assign multiple variables to represent different camera types and ensure only one type of camera is selected by apply a group constraint same as in Section 3.3.4.3.

The random parameters for the target object in computing the visibility function include the location and the orientation. An intuitive way is to evenly partition these spaces into equal size cells and use their centers as grid points, as in Figure 3.4a. Other discretization may also apply, including random sampling, stratified sampling and systematic sampling [80, ch.3] as shown in Figure 3.4b and 3.4c. In some surveillance applications, the possible location of the target object can be better generalized using trajectories. This is especially the case in traffic monitoring and surveillance for pedestrians. [25] is such an example (Figure 3.4d).

As for occlusion, our goal is to perform the worst-case analysis so that as long as the occlusion angle is less than a given β — which is the limit of mutual occlusion assuming two individuals always keep a minimum distance with each other — our solution is guaranteed to work no matter where the occlusion is. As such, a straightforward quantization of the starting position β_s of the occlusion angle will not work – an occlusion angle of β starting anywhere between grid points will occlude



(a) grid sample (b) random sample

(c) stratified (d) Paths

Figure 3.4: Different sample strategy for discretizing the object space

additional view. To simultaneously discretize the space and maintain the guarantee, we select a larger occlusion angle $\beta_m > \beta$ and quantize the starting position of the occlusion angle using a step-size of $\beta_\Delta = \beta_m - \beta$. The occlusion angles considered under this discretization will then be $\{[i\beta_\Delta, i\beta_\Delta + \beta_m) : i = 0, \ldots, N_\beta - 1\}$ where $N_\beta = \lceil (\pi - \beta_m)/\beta_\Delta \rceil$. This guarantees that any occlusion angle less than or equal to β will be covered by one of the occlusion angles. Figure 3.5 show an example of $\beta = \beta_\Delta = \pi/4$ and $\beta_m = \pi/2$.

Given a camera grid point and a target grid point, we can explicitly predict

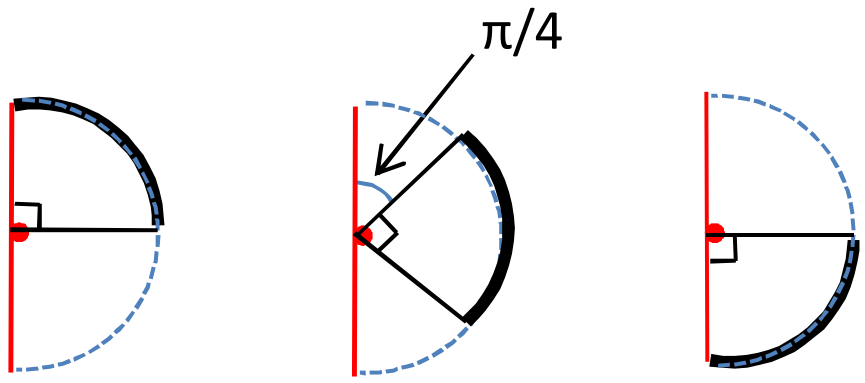


Figure 3.5: Discretization for occlusion angle. *Discretization to guarantee occlusion less than $\beta = \pi/4$ at any position will be covered in one of the above three cases:* $[0, \frac{\pi}{2}), [\frac{\pi}{4}, \frac{3\pi}{2})$ *and* $[\frac{\pi}{2}, \pi)$.

whether a target object will be visible or not to a particular camera grid point, using the visibility function in Section 4.3.2.2. These numerical values for every pair of camera and target grid points consists a "visibility matrix", which will be used in formulating our optimal camera placement problem together with the discretized camera and object space.

3.4 Optimal Camera Placement Formulation

The goal of an optimal camera placement is to identify, among all possible camera network configurations, the one that maximizes the visibility function given by (3.2). As (3.2) does not possess an analytic form, it is very difficult to apply conventional continuous optimization strategies such as variational techniques or convex programming. As such, we follow a similar approach as in [28] by finding an approximate solution over a discretization of two spaces – the space of possible camera configurations and the space of target location and orientation. The optimization problem over

the discrete spaces is formulated as a Binary Integer Programming (BIP) problem in which binary variables are used to indicate whether a camera is placed at a specific grid point and whether a target is observable at a particular location and orientation.

Binary integer programming is a very powerful tool for mathematical modeling. In this section, we present various formulations for different application requirements. Before going to the detail models, we firstly make some simple but important notations below.

- $b_j, j = 1, 2, \ldots, N_c$: is a group of binary variable indicating whether to put a camera in a specific camera grid point j. N_c is the size of the discrete camera space.

- $x_i, i = 1, 2, \ldots, N_t$: is a group of binary variable indicating whether a target object at grid point i is "observable" under a specific camera placement. N_t is the size of the discrete target object space.

- $I_b(\Lambda_j|w, T, K, \Upsilon_i), i = 1, 2, \ldots, N_t, j = 1, 2, \ldots, N_c$: as described in Equation (3.7),the "visibility matrix" with binary value indicating whether a target object at grid i is visible by camera at grid j.

3.4.1 MIN_CAM: Minimizing the number of cameras for a target visibility

MIN_CAM estimates the minimum number of cameras which can provide a mean visibility η equal to or higher than a given threshold η_t. There are two main characteristics of MIN_CAM: first, η is computed not on the discrete target space but on

the actual continuous space using Monte Carlo simulation. As such, the measurement is independent of the discretization. Furthermore, if the discretization of the target space is done with enough prior knowledge of the environment, MIN_CAM can achieve the target using very few grid points. This is important as the complexity of BIP depends greatly on the number of constraints which is proportional to the number of grid points. Second, the requirements are formulated as constraints rather than the cost function in the BIP formulation of MIN_CAM. Thus, the solution will guarantee the chosen target grid points be visible at two or more cameras. While this is useful to those applications where the requirement in the environment needs to be strictly enforced, they may inflate the number of cameras needed to capture some poorly chosen grid points. Before describing the details of how we handle this problem, we first describe the BIP formulation in MIN_CAM.

We first associate each camera grid point Υ_i in $gridC$ with a binary variable b_i such that

$$b_i = \begin{cases} 1 & \text{if a camera is present at } \Upsilon_i \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

The optimization problem can be described as the minimization of the number of cameras:

$$\min_{b_i} \sum_{i=1}^{N_c} b_i \tag{3.9}$$

subjected to the following two constraints: first, for each target point Λ_j in $gridP$, we have

$$\sum_{i=1}^{N_c} b_j \cdot I_b(\Lambda_j | w, T, K, \Upsilon_i) \geq 1 \tag{3.10}$$

This constraint represents the requirement that all targets must be visible at least one camera. As defined in Equation (3.7), $I_b(\Lambda_j|w, T, K, \Upsilon_i)$ measures the visibility of target Λ_j with respect to camera at Υ_i. In other words, Λ_j satisfying the constraint (3.10) must be in the perfect zone. Second, for each camera location (x, y), we have

$$\sum_{\text{all } \Upsilon_i \text{ at } (x,y)} b_i \leq 1 \tag{3.11}$$

These are a set of inequalities guaranteeing that only one camera is placed at any spatial location. The optimization problem in (3.9) with constraints (3.10) and (3.11) forms a standard BIP problem.

The solution to the above BIP problem obviously depends on the selection of grid points in $gridP$ and $gridC$. While $gridC$ is usually predefined according to the constraint of the environment, there is no guarantee that, as alluded to in Section 3.2, a target at a random location can be visible by two cameras even if there is a camera at every camera grid point. Thus, target grid points must be placed intelligently – target grid points away from obstacles and walls are usually easier to observe. On the other hand, focusing only on areas away from the obstacles may produce a subpar result when measured over the entire environment. To balance the two considerations, we solve the BIP repeatedly over a progressively refined $gridP$ over the spatial dimensions until the target η, measured over the entire continuous environment, is satisfied. One possible refinement strategy is to have $gridP$ started from a single grid point at the middle of the environment, and grew uniformly in density within the interior of the environment but remains at least one interval away from the boundary. If the BIP fails to return a solution, the algorithm will randomly remove half of the newly added

target grid points. The iteration terminates when the target η_t is achieved or all the newly added grid points are removed. The above process is summarized in Algorithm 1.

Input: initial grid points for cameras $gridC$ and target $gridP$, η_t, maximum grid density $maxDensity$
Output: Camera placement $camPlace$
Set $\eta = 0$, $newP = \emptyset$;
while $\eta \leq \eta_t$ **do**
  foreach Υ_i *in* $gridC$ **do**
    foreach Λ_j *in* $gridP \cup newP$ **do**
      Calculate $I_b(\Lambda_j | w, T, K, \Upsilon_i)$;
    end
  end
  Solve $newCamPlace = \text{BIP\_solver}(gridC, gridP, I_b)$;
  if $newCamPlace == \emptyset$ **then**
    if $|newP| == 1$ **then**
      break, return failure ;
    Randomly remove half of the elements from $newP$;
  else
    $camPlace = newCamPlace$;
    $gridP = gridP \cup newP$;
    $newP =$ new grid points created by halving the spatial separation;
    $newP = newP \setminus gridP$;
    Calculate η for $camPlace$ by Monte Carlo Sampling;
  end
end

Algorithm 1: MIN_CAM Algorithm

3.4.2 FIX_CAM: Maximizing the visibility for a given number of cameras

A drawback of MIN_CAM is that it may need a large number of cameras in order to satisfy the visibility of all target grid points. If the goal is to maximize the average visibility, a sensible way to reduce the number of cameras is to allow a small portion of the target grid points not being observed by two or more cameras. As long as the

target grid is dense, such "blind spots" will be rare as guaranteed by a high average visibility. FIX_CAM is the algorithm that does precisely that.

We first define a set of binary variables on the target grid $\{x_j : j = 1, \ldots, N_p\}$ indicating whether a target on the j^{th} target point in $gridP$ is visible at two or more cameras. In order to maximize the visibility, the objective function for BIP becomes,

$$\max_{b_i} \sum_{j=1}^{N_p} x_j \tag{3.12}$$

The relationship between the camera placement variables b_i's as defined in (3.8) and visibility performance variables x_j's can be described by the following constraints. For each target grid point Λ_j, we have

$$\sum_{i=1}^{N_c} b_i I_b(\Lambda_j | w, T, K, \Upsilon_i) - N_c x_j \leq 0 \tag{3.13}$$

$$\sum_{i=1}^{N_c} b_i I_b(\Lambda_j | w, T, K, \Upsilon_i) - x_j \geq 0 \tag{3.14}$$

These two constraints effectively define the binary variable x_j: if $x_j = 1$, Inequality (3.14) becomes

$$\sum_{i=1}^{N_c} b_i I_b(\Lambda_j | w, T, K, \Upsilon_i) \geq 1$$

which means that a feasible solution of b_i's must have the target visible at two or more cameras. Inequality (3.13) becomes

$$\sum_{i=1}^{N_c} b_i I_b(\Lambda_j | w, T, K, \Upsilon_i) \leq N_c$$

which is always satisfied – the largest possible value from the left-hand size is N_c corresponding to the case when there is a camera at every grid point and every target

point is observable by two or more cameras. If $x_j = 0$, Inequality (3.13) becomes

$$\sum_{i=1}^{N_c} b_i I_b(\Lambda_j | w, T, K, \Upsilon_i) \leq 0$$

which implies that the target is not visible by two or more cameras. Inequality (3.14) is always satisfied as it becomes

$$\sum_{i=1}^{N_c} b_i I_b(\Lambda_j | w, T, K, \Upsilon_i) \geq 0$$

Two additional constraints are needed to complete the formulation: as the cost function focuses only on visibility, we need to constrain the number of cameras to be less than a maximum number of cameras as follows:

$$\sum_{j=1}^{N_c} b_j \leq m \tag{3.15}$$

We also keep the constraint in (3.11) to ensure only one camera is used at each spatial location.

Unlike MIN_CAM, the feasible solution set for FIX_CAM is non-empty – no matter how dense we set the discretization, the trivial case of no target being observed will always satisfy the constraints. As such, we can simply run FIX_CAM on a fixed dense grid without any refinement of the target space. FIX_CAM is more computationally intensive than MIN_CAM as there are two constraints for each target grid point and usually a denser grid is used. Since this algorithm is more complex and requires the specification of a target number of cameras, a possible strategy is to use the MIN_CAM to estimate the approximate number of cameras and gradually reduce the number of cameras using FIX_CAM until the mean visibility falls below the target.

Alternatively, we can incorporate the minimization of the number of cameras by modifying the cost function defined in Equation (3.12) to the following:

$$\max_{b_i} \sum_{j=1}^{N_p} x_j - \sigma \sum_{i=1}^{N_c} b_i \tag{3.16}$$

where σ is a user-defined parameter for balancing the maximization of visibility and minimization of the number of cameras. Experimental results using FIX_CAM will be shown in Section 3.4.1.

3.4.3 Common Requirements in Camera Planning

Connectivity

In addition we can add constraints to model the connectivity requirement for camera network. These kind of requirement is extremely useful in wireless camera network where only proximate camera nodes can communicate between each other due to the power constraints. We firstly introduce an adjacency matrix A with each entry $a_{ij} = 1$ if camera at i and j are connected, a set variable if there is a flow from camera i to j. The flow is defined as

$$y^{ij} \leq \frac{a_{ij}}{2}(b_i + b_j)$$

The connectivity is ensured by require every selected camera has at least a flow

$$b_i \leq \sum_j y^{ij}$$

A more complex example considering communication capacity and different type of sensors can be found in [81].

Localization Error

In [27], Ercan et. al models the 2D localization error of a given target for one camera as a Gaussian distribution. Therefore, they formulate the sensor selection problem to be

$$\begin{aligned} \max.&(\frac{\alpha+1}{\sigma_x^2}+\sum_i \frac{b_i}{\sigma_{vi}})^2-(\frac{\alpha-1}{\sigma_x^2}+\sum_i \frac{b_i\cos 2\theta_i}{\sigma_{vi}})^2 \\ &-(\sum_i \frac{b_i \sin 2\theta_i}{\sigma_{vi}})^2 \\ \text{s.t.}&\sum_{i=1}^{N_c} b_i \le m \quad \text{and} \quad b_i \text{ are binary.} \end{aligned} \tag{3.17}$$

Where α, σ_x, σ_v are all pre-computed constants. σ_v is the measurement noise variance, σ_x is the localization noise prior and α is an parameter to indicate the asymmetry of prior noise in the 2 principle axes. In the original paper, a Semi-definite program approach is used taking advantage of the fact that the objective function is quadratic. However, we will show that our formulation is general enough to transform this problem into a binary integer programming.

By expansion, the objective function can be rearranged into following form,

$$\begin{aligned} &\frac{4\alpha}{\sigma_x^4}+\frac{2(\alpha+1)}{\sigma_x^2}\sum_i \frac{b_i}{\sigma_{vi}^2}+\frac{2(\alpha-1)}{\sigma_x^2}\sum_i \frac{\cos 2\theta_i b_i}{\sigma_{vi}^2} \\ &+2\sum_i\sum_j \frac{1-\cos 2\theta_i\cos 2\theta_j-\sin 2\theta_i \sin 2\theta_j}{\sigma_{vi}^2\sigma_{v_j}^2} b_i b_j \end{aligned} \tag{3.18}$$

Where the only variables are b_i's. Though the expression is not linear due to the existence of $b_i b_j$, we can linearize it by replacing $b_i b_j$ with binary variables y_{ij} and

add three sets of linear constraints,

$$-b_i + y_{ij} \leq 0$$

$$-b_j + y_{ij} \leq 0$$

$$b_i + b_j - y_{ij} \leq 1 \tag{3.19}$$

Multiple Coverage Constraint

In [29], the target is considered visible if and only if its frontal view is observed by at least k cameras. This requirement can be imposed by adding the following constraints. For each target grid point Λ_j for $j = 1, 2, \ldots, N_p$, we have

$$\sum_{i=1}^{N_c} v_{ij} b_i - (N_c - k + 1) x_j \leq k + 1 \tag{3.20}$$

and

$$\sum_{i=1}^{N_c} v_{ij} b_i - k x_j \geq 0 \tag{3.21}$$

where v_{ij} is a binary visibility matrix pre-computed for each pair of camera position Υ_i and target position Λ_j, such that $v_{ij} = 1$ indicates the camera at Υ_i can observe target at Λ_j.

Note that when we want to maximize the coverage, we can drop the constraint 3.20. However, it does not necessary lead to faster solutions.

Tracking Performance

The crucial element for tracking is occlusion handling. In the sense of camera placement or selection, it means the continuous time interval when the object is not observed should be minimized. This requirement can be represented by a set of linear

constraints for every path or neighborhood denoted as Ψ_j

$$\sum_{i \in \Psi_j} x_i \leq \eta |\Psi_j|$$

Where η is the tolerance of missing frames for the tracker and $|\Psi|$ is the size of the neighborhood.

Partial Coverage

In some computer vision applications, a simple binary visibility metric is inadequate. For instance, in facial recognition system, smaller image has higher probability to produce erroneous results than a bigger image. This require a function to "gracefully" decade from full visible status to invisible. A fuzzy model has been introduced in [82], which can be easily incorporated into a binary integer programming model. All we need is to redefine v_{ij} in Equation (3.13) to be

$$v_{ij} = \begin{cases} 0 & f(\cdot) < t_1 \\ 1 & f(\cdot) \geq t_2 \\ \frac{1}{t_2 - t_1} f(\cdot) - t_1 & t_1 \leq f(\cdot) < t_2 \end{cases}$$

where $f(\cdot)$ is an image function that takes the camera position and target position to output image size. t_1, t_2 are upper and lower threshold for a computer vision task.

Group Constraint

Group constraint is used to impose some requirement in a subset of the whole parameter space. For instance, the control of a PTZ camera can be formulated as a group constrait over binary variables with known position but unknown pose. When we already know the position of a camera but want to choose an optimal pose out of

it, we can further discretized the pose parameter and impose following constraint

$$\sum_{\text{all } \Upsilon_i \text{ at } (c_x, c_y, c_z)} b_i = 1. \tag{3.22}$$

Similar constraint can be found in [79] where a set of inequalities are used to prevent multiple cameras to be placed into one physical position.

Placement of Stereo Sensors

Binary integer programming is also used to model a stereo sensor placement problem of to minimize cost while ensuring every target has been covered by a stereo pair. We provide an alternative BIP formulation using our generic framework.

$$\begin{aligned} &\text{min.} \sum_i b_i \\ &\text{s.t. } x_k = v_{i,j}^k b_i b_j = 1 \end{aligned} \tag{3.23}$$

Where $v_{i,j}^k$ is the stereo visibility metric defined as

$$v_{ij}^k = \begin{cases} 1 & \text{when } d_1 \leq d(\Upsilon_{i,j}, \Lambda_k) \leq d_2 \\ & \text{and } \angle \Upsilon_i \Lambda_k \Upsilon_j \leq \theta \\ 0 & \text{otherwise} \end{cases}$$

Where $d(\cdot, \cdot)$ is the Euclidean distance function.

The same method as in Section 3.4.3 can be used to make the constraint linear. Similar framework can be seen in [34].

3.5 Experimental Results

In this section, we present both simulation and realistic camera network results to demonstrate the proposed algorithms. In Section 3.5.1, we show various properties of

MIN_CAM, FIX_CAM and GREEDY by varying different model parameters. In Section 3.5.2, we compare the optimal camera configurations computed by our techniques with other camera configurations.

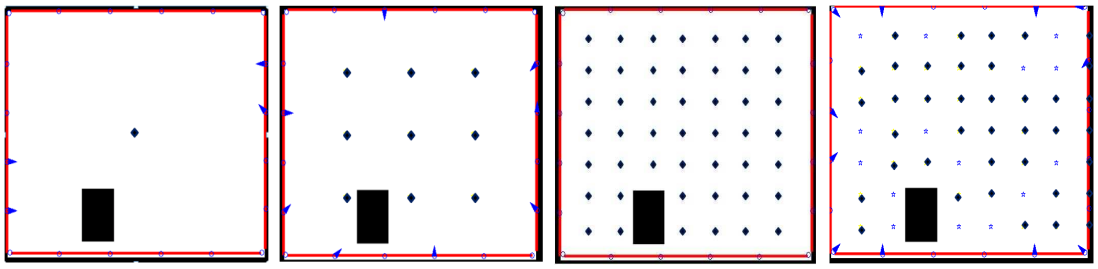
3.5.1 Optimal camera placement simulation experiments

All the simulations in this section assume a room of dimension 10m $\times$ 10m with a single obstacle and a square target with edge length $w = 20$ cm long. For the camera and lens models, we assume a pixel width of 5.6 μm, focal length of 8 cm and the field of view of 60 degrees. These parameters closely resembles the real cameras that we use in the real-life experiments. The threshold T for visibility is set to five pixels which we find to be an adequate threshold for our color-target detector.

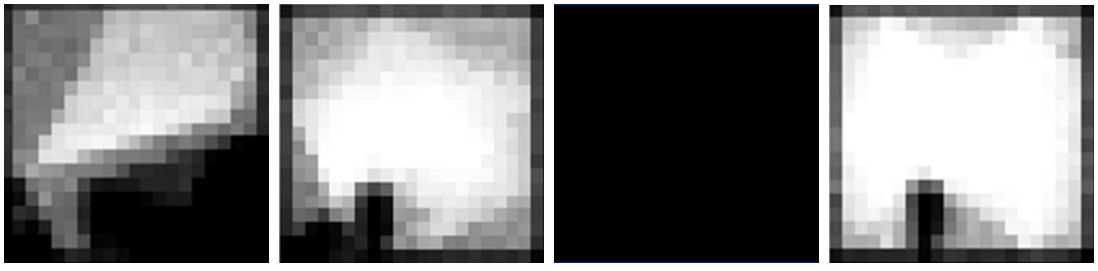
Performance of MIN_CAM

We first study how MIN_CAM estimates the minimum number of cameras for a target mean visibility η_t through target grid refinement. For simplicity, we keep all the cameras at the same elevation as the targets and assume no mutual occlusion. The target mean visibility is set to be $\eta_t = 0.90$ and the algorithm reaches this target in four iterations. The output at each iteration are shown in Figure 3.6. Figures 3.6a and 3.6e show the first iteration. Figure 3.6a shows the environment with one target grid point (black dot) in the middle. The camera grid points are restricted at regular intervals along the red boundary of the environment and remain the same for all iterations. The blue arrows indicate the output position and pose of the cameras from the BIP solver. Figure 3.6e shows the Monte-Carlo simulation results. The

mean visibility η over the environment is estimated to be 0.4743. Since it is below the target η_t, the target grid is refined as shown in Figure 3.6b with the corresponding Monte-Carlo simulation shown in Figure 3.6f. With the number of cameras increases from four to eight, η increases to 0.7776. The next iteration shown in Figure 3.6c grows the target grid further. With so many constraints, the BIP solver fails to return a feasible solution. MIN_CAM then randomly discards roughly half of the newly added target grid points. The discarded grid points are shown as blue dots in Figure 3.6d. With fewer grid points and hence fewer constraints, a solution is returned with eleven cameras. The corresponding Monte-Carlo simulation shown in Figure 3.6h gives $\eta = 0.9107$ which exceeds the target threshold and MIN_CAM terminates.



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3 (d) Iteration 4



(e) $\eta = 0.4743$ (f) $\eta = 0.7776$ (g) $\eta = 0$ (h) $\eta = 0.9107$

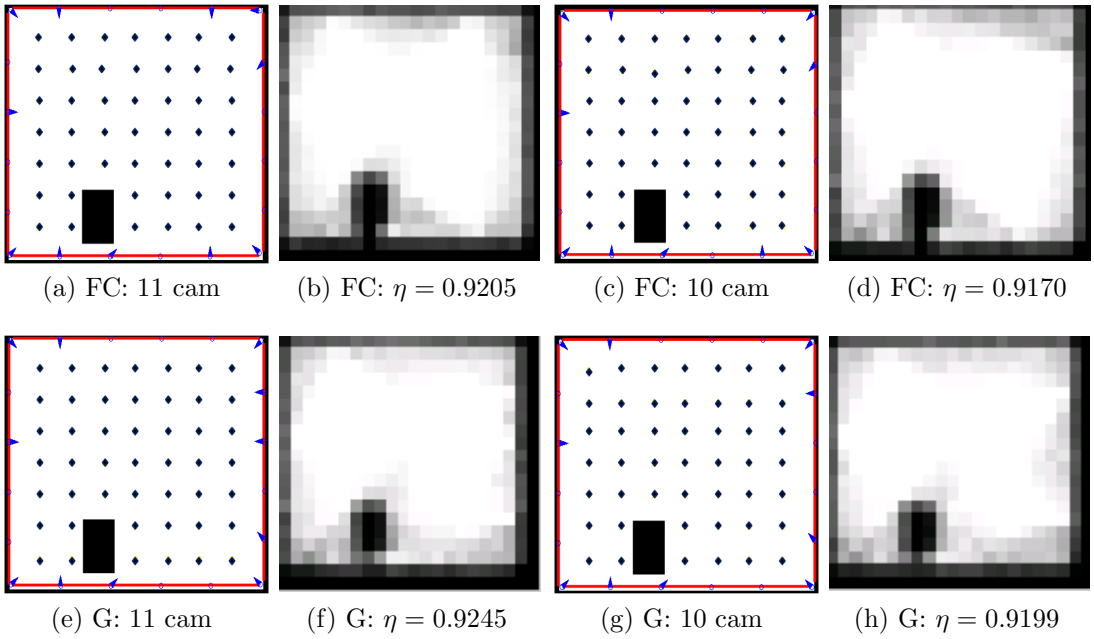
Figure 3.6: *Four iterations of MIN_CAM*

FIX_CAM versus MIN_CAM

In the second experiment, we demonstrate the difference between FIX_CAM and MIN_CAM. Using the same environment as in Figure 3.6c, we run FIX_CAM to maximize the performance with eleven cameras. The traffic model ρ_j is set to be uniform. MIN_CAM fails to return a solution under this dense grid and after randomly discarding some of the target grid points, outputs $\eta = 0.9107$ using eleven cameras. On the other hand, without any random tuning of the target grid, FIX_CAM returns a solution of $\eta = 0.9205$ and the results are shown in Figures 3.7a and 3.7b. When we reduce the number of cameras to ten and rerun FIX_CAM, we manage to produce $\eta = 0.9170$ which still exceeds the results from MIN_CAM. This demonstrates that we can use FIX_CAM to fine-tune the approximate result obtained by MIN_CAM. The camera configuration and the visibility distribution of using ten cameras are shown in Figure 3.7c and 3.7d, respectively.

GREEDY Implementation of FIX_CAM

Using the same setup, we repeat our FIX_CAM experiments using the GREEDY implementation. Our algorithm is implemented using MATLAB version 7.0 on a Xeon 2.1Ghz machine with 4 Gigabyte of memory. The BIP solver inside the FIX_CAM algorithm is based on lp_solve [83]. We have tested both algorithms using eleven, ten, nine and eight maximum number of cameras. While changing the number of cameras does not change the number of constraints, the search space becomes more restrictive as we reduce the number of cameras. As such, it is progressively more difficult to



(a) FC: 11 cam (b) FC: $\eta = 0.9205$ (c) FC: 10 cam (d) FC: $\eta = 0.9170$

(e) G: 11 cam (f) G: $\eta = 0.9245$ (g) G: 10 cam (h) G: $\eta = 0.9199$

Figure 3.7: Comparison of FIX_CAM with GREEDY algorithm. *Figures 3.7a to 3.7d show the results of using FIX_CAM (F). Figures 3.7e to 3.7h show the same set of experiments using GREEDY (G)as an approximation to FIX_CAM.*

prune the search space, making the solver resemble that of an exhaustive search. The results are summarized in Table 3.2. For each run, three numerical values are reported: the fraction of target points visible to two or more cameras which is the actual minimized cost function, the running time and the mean visibility estimated by Monte Carlo simulations. At eight cameras, GREEDY is 30,000 times faster than lp_solve but only 3% fewer visible target points than the exact answer. It is also worthwhile to point out that the lp_solve fails to terminate when we refine the target grid by halving the step-size at each dimension, while GREEDY uses essentially the same amount of time. The placement and visibility maps of the GREEDY algorithm that mirror those from FIX_CAM are shown in the second row of Figure 3.7.

Table 3.2: *Comparison between Lp_solve and greedy*

No. cameras	Lp_solve			Greedy		
	Visible targets	Time(s)	η	Visible targets	Time(s)	η
Eleven	0.99	1.20	0.9205	0.98	0.01	0.9245
Ten	0.98	46.36	0.9170	0.98	0.01	0.9199
Nine	0.97	113.01	0.9029	0.97	0.01	0.8956
Eight	0.96	382.72	0.8981	0.94	0.01	0.8761

Elevation of targets and cameras

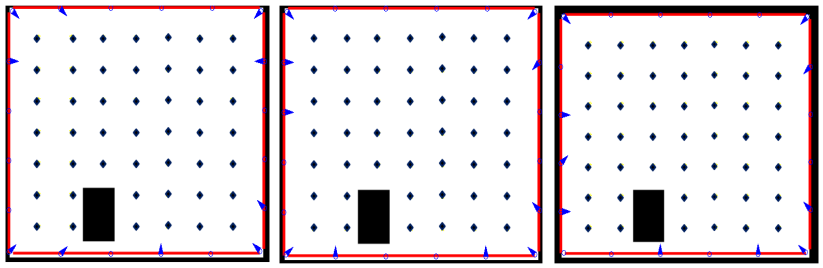
Armed with an efficient greedy algorithm, we can explore various modeling parameters in our framework. An assumption we made in the visibility model is that all the target centers are in the same horizontal plane. This does not reflect the real world due to the different height of individuals. In the following experiment, we examine the impact of the variation in height on the performance of a camera placement. Using the camera placement in Figure 3.7g, we simulate five different scenarios: the height of each person is 10 cm or 20 cm taller/shorter than the assumed height, as well as heights randomly drawn from a bi-normal distribution based on U.S. census data [84]. The changes in the average visibility are shown in Table 3.3. They range from -3.8% to -1.3% which indicate that our assumption does not has a significant impact on the measured visibiliy.

Table 3.3: *Effect of height variation on η*

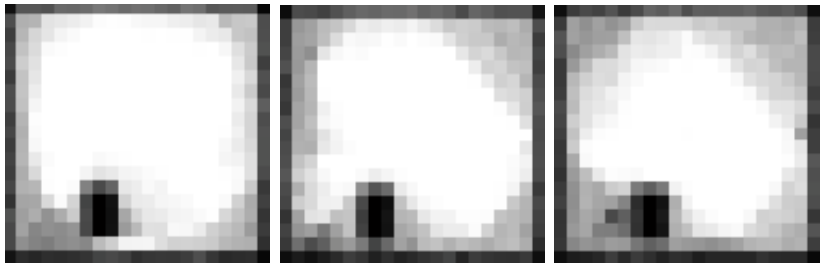
height model	+20	-20	+10	-10	Random
Change in η	-3.8%	-3.3%	-1.2%	-1.5%	-1.3%

Next, we consider the elevation of the cameras. In typical camera networks, cameras are usually installed at elevated positions to mitigate occlusion. The drawback

of the elevation is that it has a smaller field of view when compared with the case when the camera is at the same elevation as the targets. By adjusting the pitch angle of an elevated camera, we can selectively move the field of view to various part of the environment. As we now add one more additional dimension of pitch angle, the optimization becomes significantly more difficult and GREEDY algorithm must be used. Figure (3.8) shows the result for $m = 10$ cameras with three different elevations above the Γ plane on which the centers of all the targets are located. As expected, the mean visibility reduces as we raise the cameras. The visibility maps in Figures 3.8d, 3.8e and 3.8f show that as the cameras are elevated, the coverage near the boundary drops but the center remains well-covered as the algorithm adjusts the pitch angles of the cameras.



(a) 0.4m (b) 0.8m (c) 1.2m

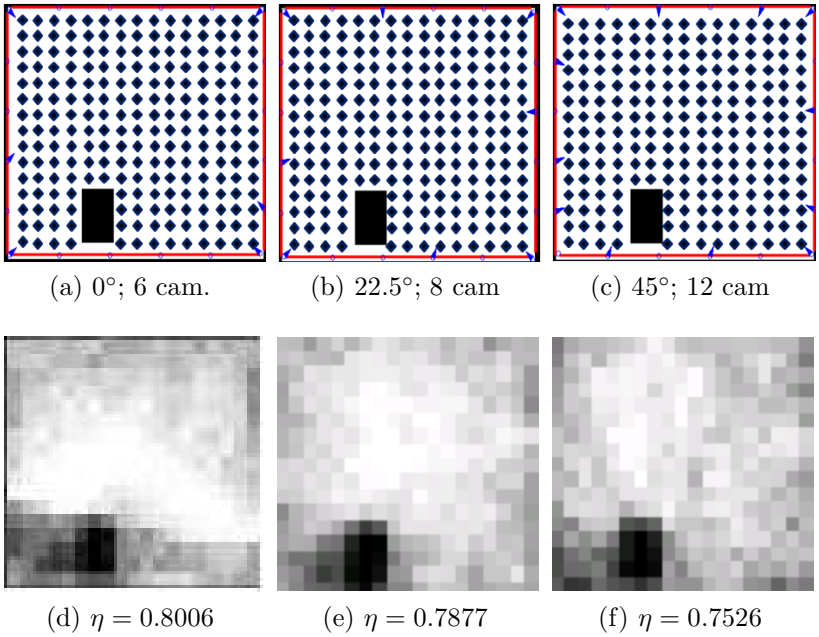


(d) $\eta = 0.9019$ (e) $\eta = 0.8714$ (f) $\eta = 0.8427$

Figure 3.8: Camera planning and Monte-Carlo simulation results. *Cameras are elevated to be 0.4, 0.8 and 1.2m above the targets.*

Mutual Occlusion

We present simulation results to show how our framework deals with mutual occlusion. Recall that we model occlusion as an occlusion angle of β at the target. Similar to the experiments on camera elevation, our occlusion model adds an additional dimension to the target grid and thus we have to resort to the GREEDY algorithm. We would like to investigate how occlusion affects the number of cameras and the camera positions of the output configuration. As such, we use GREEDY to approximate MIN_CAM by identifying the minimum number of cameras to achieve a target level of visibility. We use a denser target grid than before to minimize the difference between the actual mean visibility and that estimated by GREEDY over the discrete target grid. The target grid we use is 16×16 spatially with 16 different orientations. We set the target to be $\eta_t = 0.8$ and test different occlusion angle β at $0°$, $22.5°$ and $45°$. As explained earlier in Section 3.3, our discretization uses a slightly larger occlusion angle to guarantee worst-case analysis – we uses $\beta_m = 32.5°$ for $\beta = 22.5°$ and $\beta_m = 65°$ for $\beta = 45°$. In the Monte Carlo simulation, we put the occlusion angle at random position of each sample point. The results are shown in Figure 3.9. We can see that even with increasing number of cameras from six to eight to twelve, the resulting mean visibility suffers slightly when the occlusion angle increases. Another interesting observation from the visibility maps in Figures 3.9d, 3.9e and 3.9f is that the perfect region, indicated by the white pixels, dwindles as occlusion increases. This is reasonable because it is difficult for a target to be visible at all orientation in the presence of occlusion.



(a) 0°; 6 cam. (b) 22.5°; 8 cam (c) 45°; 12 cam

(d) $\eta = 0.8006$ (e) $\eta = 0.7877$ (f) $\eta = 0.7526$

Figure 3.9: Comparing different occlusion angles. *As the occlusion angle increases from* 0° *in Figure 3.9a to* 22.5° *in Figure 3.9b and* 45° *in Figure 3.9c, the required number of cameras increases from 6 to 8 and 12 when using GREEDY to achieve a target performance of* $\eta_t = 0.8$. *Figure 3.9d to Figure 3.9f are the correspondent visibility maps.*

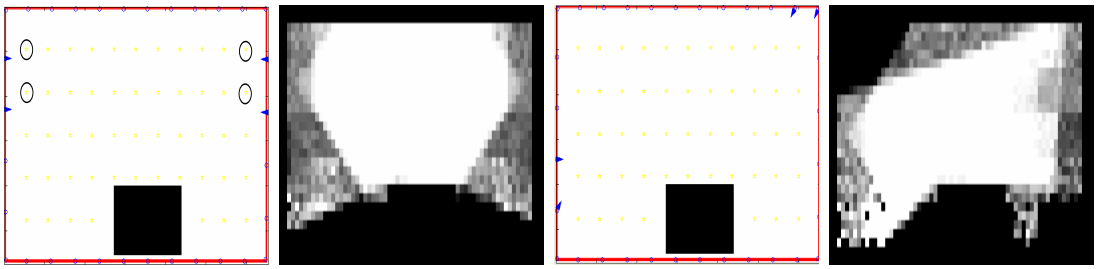
Realistic Occupant Traffic Distribution

In this last experiment, we show how one can incorporate realistic occupant traffic patterns into the FIX_CAM algorithm. All experiments thus far assume an uniform traffic distribution over the entire target space – it is equally likely to find a person at each spatial location and at each orientation. This model does not reflect many real-life scenarios. For example, consider a hallway inside a shopping mall: while there are people browsing at the window display, most of the traffic flows from one end of the hallway to the other end. By incorporating an appropriate traffic model, the performance should be improved under the same resource constraint. In the FIX_CAM framework, a traffic model can be incorporated into the optimization by

using non-uniform weights ρ_j in the cost function (3.12).

In order to use a reasonable traffic distribution, we employ a simple random walk model to simulate a hallway environment. We imagine that there are openings on the either sides of the top portion of the environment. At each of the target grid point, which is characterized by both the orientation and the position of a walker, we impose the following transitional probabilities: a walker has a 50% chance of moving to the next spatial grid point following the current orientation unless it is obstructed by an obstacle, and has a 50% chance of changing orientation. In the case of changing orientation, there is a 99% chance of choosing the orientation to face the target grid point closest to the nearest opening while the rest of the orientations share the remaining 1%. At those target grid points closest to the openings, we create a virtual grid point to represent the event of a walker exiting the environment. The transitional probabilities from the virtual grid point back to the real target points near the openings are all equal. The stationary distribution ρ_j is then computed by finding the eigenvector with eigenvalue one of the transitional probability matrix of the entire environment[85][ch.11.3].

Figure 3.10a shows this hallway environment. The four hollow circles indicate the target grid points closest to the openings. The result of the optimization under the constraint of using four cameras is shown in Figure 3.10b. Clearly the optimal configuration favors the heavy traffic hallway area. If the uniform distribution is used instead, we obtain the configuration in Figure 3.10c and the visual map in Figure 3.10d. The average visibility drops from 0.8395 to 0.7538 as there is a mismatch of the traffic pattern.



(a) Random Walk (b) $\eta = 0.8395$ (c) Uniform (d) $\eta = 0.7538$

Figure 3.10: Random walk model for camera placement. *Figures 3.10a and 3.10b use the specific traffic distribution for optimization and obtain a higher η as compared to using an uniform distribution in figures 3.10c and 3.10d.*

3.5.2 Comparison with other camera placement strategies

In this section, we compare our optimal camera placements with two different placement strategies. The first one is uniform placement – assuming that the cameras are restricted along the boundary of the environment, the most intuitive scheme is to place them at regular intervals on the boundary, each pointing towards the center of the room. The second one is based on the optimal strategy proposed in [28].

To test the differences in visibility models, it is unfair to use Monte-Carlo simulations which use the same model as the optimization. As a result, we resort to virtual environment simulations by creating a virtual 3-D environment that mimics the actual 10m×10m room used in Section 3.5.1. We then insert a random-walking humanoid wearing a red target. The results are based on the visibility of the target in two or more cameras. The cameras are set at the same height as the target and no mutual occlusion modeling is used. The optimization is performed with respect to a fixed number of cameras. To be fair to the scheme in [28], we run their optimization formulation to maximize the visibility from two cameras. The measurements of η for

the three schemes with the number of cameras varied from five to eight are shown in Table 3.4. Our proposed FIX_CAM performs the best followed by the uniform placement. The scheme in [28] does not perform well as it does not take into account the orientation of the target. As such the cameras do not compensate each other when the target is in different orientations.

Table 3.4: *η measurements among the three schemes using virtual simulations*

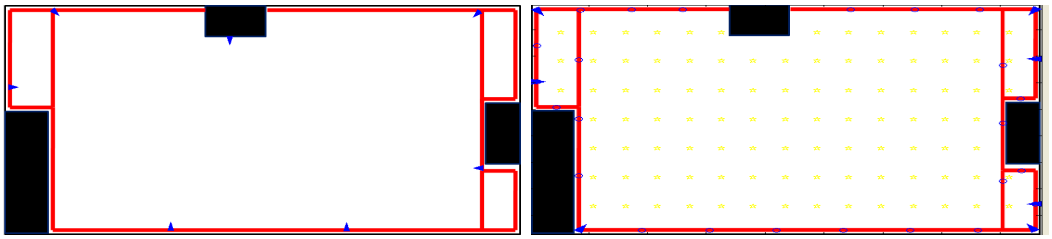
Number of cameras	FIX_CAM	[28]	Uniform Placement
5	0.614 ± 0.011	0.352 ± 0.010	0.522 ± 0.011
6	0.720 ± 0.009	0.356 ± 0.010	0.612 ± 0.011
7	0.726 ± 0.009	0.500 ± 0.011	0.656 ± 0.010
8	0.766 ± 0.008	0.508 ± 0.011	0.700 ± 0.009

We are, however, surprised by how close uniform placement is to our optimal scheme. Thus, we further test the difference between the two with a real-life experiment that incorporates mutual occlusion. We conduct our real-life experiments indoor in a room of 7.6 meters long, 3.7 meters wide, and 2.5 meters high. There are two desks and a shelf along three of the four walls. Seven Unibrain Fire-i400 cameras at elevation of 1.5 meters with Tokina Varifocol TVR0614 lens are used. Since they are variable focal-length lens, we have set them at a focal length of 8mm with a vertical field of view of 45° and horizontal field of view of 60°. As the elevation of the cameras is roughly level with the position of the targets, we have chosen a fairly large occlusion angle of $\beta_m = 65°$ in deriving our optimal placement. Monte-Carlo results between the uniform placement and the optimal placement are shown in Figure 3.11. For the virtual environment simulation, we replace the desks and the shelf with tables and teapot, insert three randomly walking humanoids and capture 250 frames

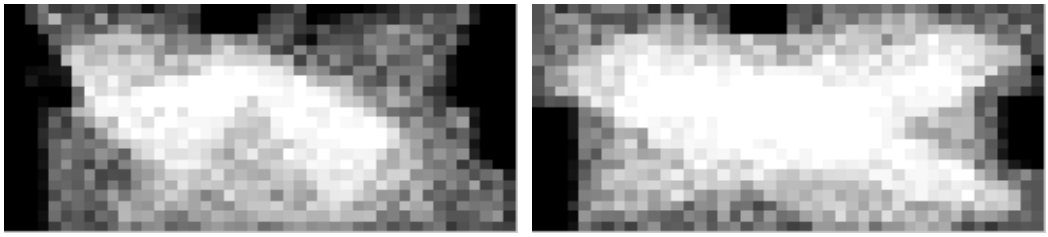
for measurement. For the real-life experiments, we capture about two minutes of video from the seven cameras, again with three persons walking in the environment. Figures 3.12 and 3.13 show the seven real-life and virtual camera views from both the uniform placement and optimal placement respectively. As shown in Table 3.5, the optimal camera placement is better than the uniform camera placement in all three evaluation approaches. The three measured η's for the optimal placement are consistent. The results of the uniform placement have higher variation most likely due to the fact that excessive amount of occlusion makes detection of color targets less reliable.

Table 3.5: *η measurements between uniform and optimal camera placements*

Methid	MC Simulations	Virtual Simulation	Real-life Experiments
Uniform	0.3801	0.4104 ± 0.0153	0.2335 ± 0.0112
Optimal	0.5325	0.5618 ± 0.0156	0.5617 ± 0.0121



(a) Uniform placement,$\eta = 0.3801$ (b) Optimal placement,$\eta = 0.5325$



(c) Uniform placement: $\eta = 0.3801$ (d) Optimal placement: $\eta = 0.5325$

Figure 3.11: *Camera placement in a real camera network*

Figure 3.12: *Seven camera views from uniform camera placement*

Figure 3.13: Seven camera views from optimal camera placement

Chapter 4 Approximate Techniques in Solving the Camera Planning Problems

While the theoretical foundation of optimal camera placement has been studied for decades, its practical implementation has recently attracted significant research interest due to the increasing popularity of visual sensor network. The discrete camera placement problem is NP-hard and many approximate solutions have been independently studied. The goal of this chapter is to provide a comprehensive framework in comparing the merits of these techniques. We consider two general classes of camera placement problems and adapt some of the most commonly used approximation techniques in solving them. The accuracy, efficiency and scalability of each technique are analyzed and compared in depth. Extensive experimental results are provided to illustrate the strength and weakness of each method.

To the best of our knowledge, there is no prior work covering the entire spectrum of approximation algorithms from Greedy, Markov Chain Monte Carlo (MCMC) to various relaxation techniques. Our key contribution is to provide not only a generic formulation of the camera placement problem but also detailed approaches to adapt the formulation to various approximation schemes. Our adaptations to Simulated Annealing (SA) and Semi-Definite Program (SDP) are novel. We demonstrate that greedy approach and its variants can obtain a good crude estimation. MCMC approaches are more complex but still return good solutions even in complex problems. Linear Programming (LP) and (SDP) relaxations are most complex but they can

provide good performance bounds.

4.1 Camera placement in general

This section we recap the general camera placement formulation mentioned in last chapter. Although there is a myriad of camera placement problems in the literature, the fundamental objectives of these problems almost always fall into two broad categories which we refer to as the MIN and FIX problems. The goal of the MIN problems is to minimize the number of cameras such that a target coverage rate p can be achieved subject to other constraints. The goal of the FIX problems is to maximize the coverage of targets subject to fixed number of cameras m and other application specific constraints.

Both problems can be tackled in the following fashion. First, the space of possible camera configurations, including locations, yaw, pitch angles, and camera types can be converted into discrete points by either a random sampling [28] or uniform discretization [29, 30]. The target space of the camera network can also be discretized into a finite space, which can be the possible 2-D [36] or 3-D [25] object positions, object orientations[86], motion paths [25] or even a combination of all the above spaces.

We denote the discretized camera space as $\{\Upsilon_i : i = 1, \ldots, N_c\}$ and the target space as $\{\Lambda_j : j = 1, 2, \ldots, N_p\}$. We then define two sets of binary variables $\{b_i : i = 1, \ldots, N_c\}$ and $\{x_j : j = 1, \ldots, N_p\}$ on the two spaces respectively. So $b_i = 1$ for $i = 1, \ldots, N_c$ indicates that a camera is placed or selected at Υ_i, and $x_j = 1$ for $j = 1, \ldots, N_p$ indicates an object at Λ_j can be observed under a given camera

placement plan. Using these variables, MIN and FIX can be formulated as follows:

$$\text{MIN:} \quad \text{minimize} \sum_{i=1}^{N_c} b_i$$
$$\text{given} \sum_{j=1}^{N_p} x_j \geq p \cdot N_p \ \text{ and } \ x_j, b_i \text{ are binary,} \tag{4.1}$$

$$\text{FIX:} \quad \text{maximize } f(x_1, x_2, \ldots x_{N_p})$$
$$\text{given} \sum_{i=1}^{N_c} b_i \leq m \ \text{ and } \ x_j, b_i \text{ are binary.} \tag{4.2}$$

where $f(x_1, \ldots, x_{N_p})$ is an application-specific real-valued function that measures the coverage of the network. A simple example of $f(x_1, \ldots, x_{N_p})$ would be $\frac{1}{N_p} \sum_{j=1}^{N_p} x_j$ but a more sophisticated metric that considers other factors can also be used. Additional application-specific constraints can also be added. We assume that all constraints are linear in x_j's and b_i's. This assumption is not overly restrictive as there are general strategies to convert nonlinear constraints into linear ones as we seen in Section 4.3.4.

4.2 Approximation methods

The number of variables in camera placement problems is directly proportional to the volume of the search space and is typically very large even for simple environments. Although there is optimization software capable of solving integer program (IP) problems, it is in general impractical or even impossible to obtain an exact solution for any reasonable-size camera placement problem. In this section we investigate several approximation methods for camera placement problems and we will show how close approximated solutions are to the exact solutions by simulations in Section 4.3.

4.2.1 Greedy method

The greedy method is probably the most intuitive method to solve these kind of problems. The basic idea is that instead of seeking a global optimum by checking all possible configurations, we choose one camera that optimizes the objective value at each step. The advantages of the greedy algorithm include a simple implementation and tremendous efficiency — most greedy algorithms have $O(n)$ complexity instead of $O(n^k)$ by using an exhaustive search. A generalized greedy algorithm for the camera placement problems in Algorithm 2.

Input: Initial grid points for cameras Υ and targets Λ, feasible sets defined by other constraints S, the target mean visibility p, maximum number of cameras m.
Output: Camera placement *camPlace*
Set $U = \Upsilon$, $V = \emptyset$, $W = \Lambda$, $camPlace = \emptyset$;
while $|V| < p \cdot |\Lambda|$ *for MIN or* $|camPlace| < m$ *for FIX* **do**
  $c =$ the camera grid point in U that maximizes the number of visible target grid points in W;
  if $camPlace \cup \{c\} \in S$ **then**
    $camPlace = camPlace \cup \{c\}$;
    $T =$ subset of *grid* visible by k in *camPlace*;
    $V = V \cup T$;
    $W = W \setminus T$;
  $U = U \setminus c$;
end
Output *camPlace*

Algorithm 2: *GREEDY: a greedy search camera placement algorithm.*

In fact, the greedy algorithm has deeper theoretical motivations than intuition. In combinatorial optimization, there is a well-studied class of problems known as "*set cover*" [87] defined as follows: Given a finite set X and a family (F) of subsets of X, a cover is a subfamily of sets whose union is X. The *set covering* optimization problem is to find a covering which uses the fewest sets.

Feige has shown in [88] that the greedy algorithm is the best polynomial-time approximation for the set covering problem by the worst-case analysis if $P \neq NP$. The worst-case performance is $\ln|X| + 1$. We can show that MIN is a set covering problem if the target coverage p is one and no other constraint is imposed. It can be achieved by setting the observation space Λ as the finite set X and each camera point Υ_i as an element in $\mathcal{F}$ that consists of all of the points in X observable by the camera. Even for the more complex MIN problems which require $p < 1$ and/or each target point observed by more than one camera, there has been much work showing that the greedy algorithm still has the same lower bound [89, 90] with little modifications.

Note that when using Algorithm 2 to solve the FIX problem, we simply assume the objective function is $\sum_{i=1}^{N_c} x_i$. Therefore it makes sense to approximate the optimum by selecting the cameras with maximized coverage at given step. This restricted version of FIX problem without any other constraints is called "*max k cover*" in [88], which was shown to admit the greedy algorithm as an efficient approximation.

However, the greedy algorithm is not the only possible approximation for the generic FIX problem. In some applications, the objective functions have different forms which does not allow meaningful interpretation when we select one camera at one step. In [29], the objective function is to maximize the number of target points observable by k cameras. When selecting the first $k-1$ cameras, no local optimum can be calculated. For MIN, we can simply split each target into k instances and eventually there will be $p \cdot k \cdot |\Lambda|$ instances covered. For FIX, the algorithm may terminate when no target is covered k times, thus leading to an objective value equal to 0. Also, when the constraints of the problem become complicated, the

greedy algorithm wastes time searching for feasible solutions, leading to an inefficient algorithm. For instance, the problem in [34] has the "sensor matching constraints" in addition to the covering constraints to ensure that enough cameras are assigned to the targets. In [30] and [29], the camera pose is also discretized so that another constraint is needed to ensure no two cameras in the same position can be selected.

Since the greedy approach already provides a good approximation for MIN, the following discussions will focus on FIX. Note that even for the MIN problems which do not admit a greedy solution, we can always iteratively apply a solver for FIX for different number of cameras and search for the minimum one that satisfies the target coverage rate.

4.2.2 Heuristics

As mentioned in the previous section, the objective functions of some camera placement problems cannot be computed by adding one camera at a time. Nevertheless, we can still follow the idea of finding local maximum/minimum at each iteration from a different perspective. The following example is a popular greedy heuristic (Algorithm 3) for FIX.

In this greedy heuristic, we have a well-defined objective function in each iteration. However, there is no sensible way to choose one set of initial $camPlace$ over another. We therefore choose a random initialization. The use of randomness inspires us to look into another set of powerful tools — random sampling.

Input: Initial grid points for cameras Υ and targets Λ, feasible sets defined by other constraints S, objective function $f(\cdot)$, max iterations N and the maximum number of cameras m.
Output: *camPlace*
Set $U = \Upsilon$, *camplace* = m random cameras;
$W = U \setminus camPlace$;
for $i = 0$; $i < N$; $i = i + 1$ **do**
  select a pair $(b, c), b \in camPlace, c \in W$ maximize $f(\cdot)$ if exchange with each other;
  if $f(camPlace) \geq f(camPlace \cap \{c\} \setminus \{b\})$ **then**
    Break;
  if $(camPlace \cap \{c\} \setminus \{b\}) \in S$ **then**
    $camPlace = camPlace \cap \{c\} \setminus \{b\}$;
    $W = U \setminus camPlace$;
  $W = W \setminus c$;
end
Output *camPlace*

Algorithm 3: *Greedy Heuristic search for camera placement algorithm.*

4.2.3 Sampling methods

Although the deterministic greedy algorithm is very efficient, we cannot improve the result once a local optimum is achieved due to its deterministic nature. Random sampling methods allows a definitive advantage over deterministic approaches — *one can always improve the results by sampling more points from the distribution.*

The simplest version of a random sampler is to uniformly sample points from the camera space. We can terminate the algorithm when a good enough solution is obtained or the maximum number of iterations is reached. Due to the large search space, it is often hard for this approach to sample even a near-optimal solution in a reasonable running time. As such, this naive version of random sampling is rarely useful.

A better random sampling scheme should relate the objective value of the sampled point to the probability of it being sampled. By assigning a higher probability to

sampled points with better objective values, we have a higher chance to sample points from the distribution that are close to the global optimum.

We denote S as the set of all possible combinations of cameras subject to all constraints, $B_i = [b_0\ b_1\ \ldots b_{N_c}] \in S$ as one specific combination — a point in the search space (the camera space). We also denote $f(\cdot)$ as the objective function. The ideal probability for sampling should be

$$P(B_i) = \frac{f(B_i)}{\sum_{j \in S} f(B_j)}. \tag{4.3}$$

In order to calculate $P(B_i)$ in Equation (4.3), we need to evaluate every B_i which is as complex as performing an exhaustive search. Here we offer two effective schemes to solve this problem.

Algorithm 4 assumes different camera positions are independent from each other and the sampling probability at each camera position is directly proportional to the number of target positions it can observe. The first assumption provides an effective mean to focus on one camera at a time and the second assumption naively relates the overall objective value to coverage of a single camera. While these two assumptions provide a very crude approximation to Equation (4.3), they provide far better samples than uniform distribution and admit very efficient implementation.

We will show in Section 4.3 that Algorithm 4 provides decent results with complexity comparable to the greedy approach. On the other hand, the assumptions are very strong and are certainly not applicable in many situations. To cope with arbitrary probability functions, the most general approach is to use Monte Carlo sampling and its many variant. Here we adopt the Metropolis algorithm [91, ch.5] to

Input: Same as in Algorithm 3.
Output: *camPlace*
Set $U = \Upsilon$, $camPlace = \emptyset$;
for $i = 1, \ldots N_c$ **do**
  Calculate $P(b_i = 1)$ by aggregating the number of targets it can observe
end
$camPlace = \emptyset$, $best = 0$, $W = U \setminus camPlace$;
for $i = 0$; $i < N$; $i = i + 1$ **do**
  for $i = 1, \ldots m$ **do**
    Sample one c from W according to P;
    $cam = cam \cap c$, $W = W \setminus c$;
  end
  if *(cam* $\in S$ *and* $f(cam) > best$ **then**
    $camPlace = cam$, $best = f(cam)$;
end
Output *camPlace*

Algorithm 4: *Random sampler based on marginal distribution.*

improve tracking the probability of each point in the search space without calculating the normalization factor.

The traditional Metropolis algorithm starts from an initial setup, (1) makes a small but random perturbation, (2) calculates the gain of this perturbation, and (3) decides whether we accept the perturbation by sampling a random number and comparing with the gain. By using a perturbation and comparing the probabilities of the two setups, we no longer need to calculate the normalization factor. Algorithm 5 is the adapted algorithm for FIX.

In order to conform to the notations typically used in the Metropolis algorithm, we redefine the probability function in Equation (4.3) as the following:

$$P(B_i) = \frac{\exp \log f(B_i)}{\sum_{j \in S} f(B_j)}. \tag{4.4}$$

Thus the gain of a perturbation becomes $\log f(B_i') - \log f(B_i) = \log f(B_i')/f(B_i)$.

Input: Same as in Algorithm 3
Output: camera placement $camPlace$
Set $U = \Upsilon$, $cam = m$ random cameras;
$W = U \setminus cam$, $best = 0$, $i = 0$;
for $i = 0$; $i < N$; $i = i + 1$ **do**
    $b =$ randomly select one camera in cam;
    $c =$ randomly select one camera in W;
    $cam' = cam \setminus b \cap c$;
    if $cam' \in S$ **then**
        $\Delta h = f(cam')/f(cam)$;
        Draw random number u from uniform $(0, 1)$ distribution;
        if $\log u \leq \min(\Delta h, 1)$ **then**
            $cam = cam'$;
            if $f(cam) > best$ **then**
                $camPlace = cam'$, $best = f(cam)$;
        $W = W \setminus c$;
end
Output $camPlace$

Algorithm 5: *Metropolis sampling for FIX.*

Algorithm 5 is very similar to Algorithm 3 except for a simple change in sampling strategy – instead of always exchanging with the camera that maximizes the objective function, we choose a random candidate for exchanging. If the objective value of the random candidate is bigger than the original, then we will make the change. Otherwise, we may still make the change according to the probability calculated by the gain. Such a sampling strategy allows our algorithm to move away from local maximum and explore the rest of the search space. Also, the algorithm can be adapted by changing the perturbation range: we can choose to exchange more cameras at a time (larger perturbation) or only allow switching with cameras nearby (smaller perturbation).

The crucial step of Algorithm 5 is that we relate the possibility of a point sampled

from the distribution setup with its objective value. It is obvious that this relationship does not need to be linear. Further more, we can see that this relationship does not need to be static. There is an extensive literature on how to change this relationship to get a faster rate of landing on an optimum point, known as "simulated annealing" [92].

In order to adopt "simulated annealing" for FIX, we add another variable T to our probability function in Equation 4.4:

$$P(B_i) = \frac{\exp \log f(B_i) \cdot T}{Z}, \tag{4.5}$$

where Z is a normalization factor which can be ignored in Metropolis sampling. When we change T, we can control the probability of jumping to a point with smaller performance. When T is big the exchange is very frequent, allowing us to explore more in the search space; when T is small, we focus on searching for the maximum. As such, a simulate-annealing scheme usually starts from a high T to run a Metropolis sampling scheme and decreases T until the objective value does not change over time. The algorithm is summarized in Algorithm 6.

Input: Inputs in Algorithm 3, an initial temperature t_s, an ending temperature t_e and cooling function $f_c(t)$.
Output: Camera placement *camPlace*
Set cam = randomly chose m cameras;
$t = t_s$;
while $t > t_e$ **do**
  $[cam, bestPlace] = MetSampling(cam, N, t)$;
  if $f(bestPlace) > f(camPlace)$ **then**
    $camPlace = bestPlace$;
  $t = f_c(t)$;
end
Output *camPlace*

Algorithm 6: *Simulated Annealing algorithm for FIX.*

The cooling function $f_c(t)$ is a custom function to decrease the temperature. Usually it is chosen as a linear or logarithmic decreasing function. The function *MetSampling* is essentially Algorithm 5 using the customized probability function defined in Equation (4.5).

4.2.4 LP and SDP relaxation

A significant drawback of sampling techniques is that it may take many iterations for the algorithm to converge. Even after convergence, the algorithm provides little clue on how close the resulting approximation is when compared with the true optimal solution.

One possible solution is to relax the original formulation by replacing the binary constraints with $0 \leq x_i \leq 1$ and $0 \leq b_i \leq 1$. As a result, we will get a linear programming formulation and by solving it we may get a fractional solution instead of a binary one. The objective from the LP relaxation provides an upper bound of the original problem [93, ch.3]. Also, we can get an approximated solution by designing some rounding scheme to obtain the binary solution.

However, the gap between the LP relaxation and original BIP — called the integrality gap — is still unknown. Various methods can be used to reduce the integrality gap by adding more constraints [94, 95, 96].

Note for any binary variable x, an equivalent constraint can be given as $x(x-1) = 0$. It is known that SDP can be used to approximate this constraint better than LP (see [97] and references within). In this chapter, we adopt the "Lift and Project" method proposed by Lovász and Schrijver [94].

We generalize MIN and FIX into standard equation in Equation (4.6).

$$\begin{aligned} &\text{Minimize } -\mathbf{c}^T\mathbf{x} \\ &\text{s.t. } A\mathbf{x} \leq b \text{ and } x_i(x_i - 1) = 0, \end{aligned} \tag{4.6}$$

where $\mathbf{x}, \mathbf{b}, \mathbf{c}$ are column vectors and A is a matrix. The inequality constraint in (4.6) applies to each dimension. The "Lift and Project" process works as follows:

1. Define a variable matrix $Y = \{y_{i,j} | i, j = 0, 1, 2, \ldots n\}$, where n is the length of vector $\mathbf{x}$.

2. Replace each constraint $A_i\mathbf{x} \leq b_i$ with a set of constraints $A_j\mathbf{x} \cdot x_j \leq b \cdot x_j$ and $A_j\mathbf{x} \cdot (1 - x_j) \leq b \cdot (1 - x_j), j = 1, 2, \ldots n$.

3. Replace each instance of x variable with y such that $x_i x_j = y_{i,j}, x_i x_i = x_i = y_{i,i} = y_{0,i}$ for all i, j.

4. Replace binary constraints $x_i(x_i - 1) = 1$ with a constraint forcing Y to be positive semi-definite or $Y \succ 0$.

5. Solve the SDP problem of Y and recover $x_i = y_{0,i}$.

It is shown in Section 4.3 that the "Lift and Project" process provides a much tighter bound than LP relaxation. In fact, we can get an even tighter relaxation by continuing to raise the dimension of variables. In [97], Laurent analyzed and compared three different hierarchical methods to obtain a series SDP relaxations of the $0 - 1$ problem. However, in practical camera placement problems, the number of variables becomes large. Conducting more than one round of SDP relaxation will inevitably run into memory issues.

4.3 Simulation results

We propose three sets of experiments to illustrate the strength and weakness of each proposed method. Firstly, we compare various fast and simple algorithms on a problem with a simple topology. Then we compare them with more sophisticated algorithms on complex environments. Last but not least, we apply the LP and SDP relaxations on a small example to see how the SDP relaxation dramatically reduces the integrality gap.

For comparison, we only use the constraints in Equations (3.13), (3.14) and (3.22) with $k = 2$. All experiments were conducted on a Duo core 2.8 GHz CPU with 3.2 GB RAM, with most code written in C linked to Matlab. The IP solver we used was in [98] and SDP solver we used was SDPA [99].

4.3.1 Environment with simple topology

In this section, we test our algorithms on a simple 2D square environment to place number of cameras range from 2 to 8 as in Figure 4.1a. The blue hollow circles are discretized camera grid positions and yellow solid stars are target grid positions. The blue arrows are the placed cameras. Here we have 28 camera positions and 49 tag positions. Each position is further divided into 8 grid points to represent different orientations. The total numbers of variables are $N_c = 192$ for cameras and $N_p = 392$ for targets.

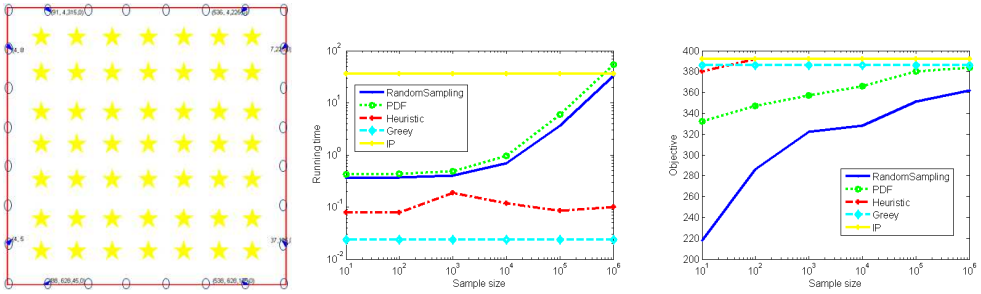
We first compare the running time for different algorithms and different sample sizes in Figure 4.1b. In Figures 4.1c – 4.1f, we compare the results for different

algorithms when the number of cameras varies. From those comparisons, we can make following observations: (1) When the number of cameras is sufficiently large, the greedy algorithm has good approximation of IP solution with a fraction of the running time. However, when the number of cameras is small, the greedy algorithm provides much worse results due to its complete overlook of the combinatorial characteristics of the problem; (2) the sampling techniques can trade off performance with computational time; (3) using elements sampled from densities derived from the objective function significantly outperforms those from uniform random sampling; (4) the greedy heuristics generally out-performs other approximation methods. However, it can still be trapped in a local optimum regardless of the sample size. We will see this disadvantage will incur big penalty when the environment becomes more complex.

4.3.2 Metropolis sampling and simulated annealing on complex problems

As we can see above, for small and simple problems we can choose from either the greedy algorithm (Algorithm 2), a greedy heuristics or sampling based on marginal distribution (Algorithm 4). Furthermore, these problems can also be solved by a standard IP solver in a reasonable running time. Now, we begin to look at much more complex environments.

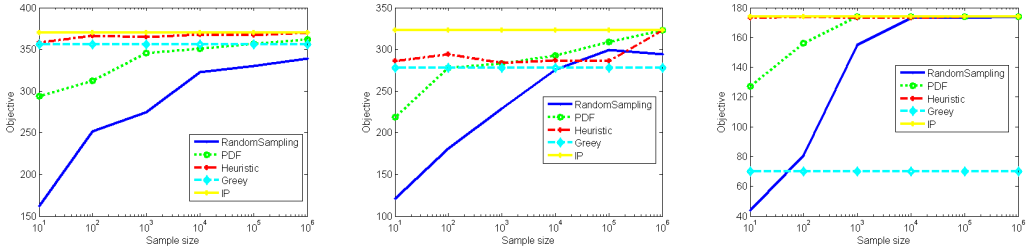
In Figure 4.2 we show two complex environments generated by our camera placement GUI to place 8 cameras. We present the performances of IP solver, the greedy algorithm, Metropolis sampling, and simulated annealing (SA) approach in Table 4.1.



(a) problem topology and optimal placement for 8cameras cameras

(b) Time comparison for 8 cameras

(c) objective comparison for 8 cameras

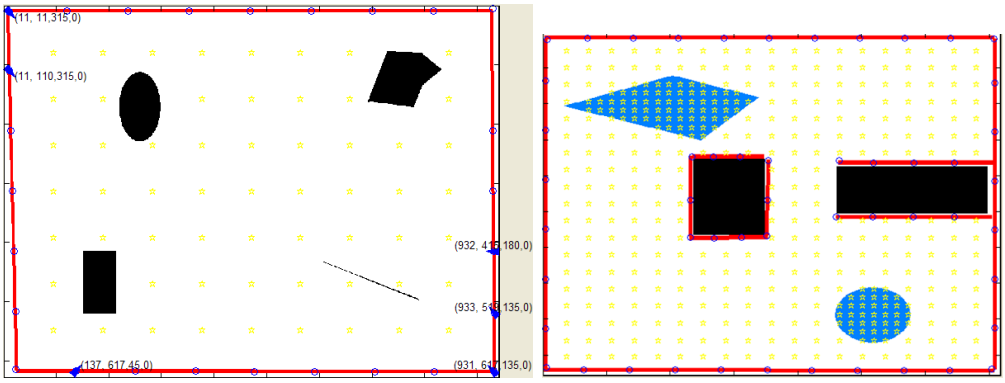


(d) objective comparison for 6 cameras

(e) objective comparison for 4 cameras

(f) objective comparison for 2 cameras

Figure 4.1: Performance comparison of four approximation algorithms



(a) Environment 1

(b) Environment 2

Figure 4.2: [Two complex topologies for algorithm comparison. *Black objects are obstacles and blue areas are secured areas with grid density 4 times higher than surroundings.*

We can conclude that both sampling algorithms are highly efficient when compared with the IP solver. We can further see the change of temperatures plays an

Table 4.1: Comparison of two Monte Carlo sampling methods with other algorithms.

	Environment 1		Environment 2	
	Objective	Time (s)	Objective	Time (s)
IP	353	1552.4	2336	101320.6
Greedy	339	0.002014	2164	0.1362
Heuristic	344	0.0297	2029	0.440203
Metropolis	352	0.6784	2290	1.109044
SA	350	1.957046	2336	7.739528

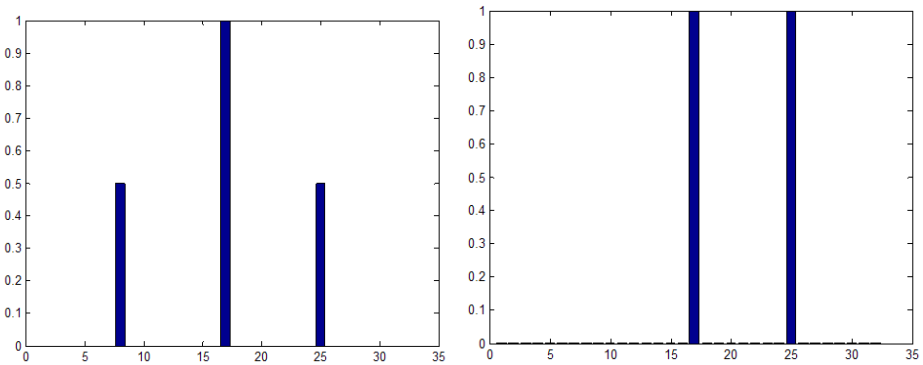
important role in escaping the local optimums and exploring the entire search spaces.

4.3.3 LP relaxation and SDP relaxation

At last, we show the effectiveness of using SDP on relaxation on a simple and a moderate complex environments. The results are shown in Table 4.2. We can see the SDP relaxation always gives a tighter bound comparing with LP relaxation. We visualize the camera grid variables for the simple topology in Figures 4.3a and 4.3b, with the variable indexes in the x axis and values in the y axis. We can see the SDP relaxation gives results closer to the binary with smaller objective value. In fact, in this particular example, the SDP relaxation solution coincides with the IP solution.

Table 4.2: Comparison of the objective values of SDP and LP relaxation

	56 Grids, 2 Cameras	268 Grids, 8 Cameras
LP	11.5	32.25
SDP	11	31.34
Optimal	11	31



(a) LP objective = 11.5 (b) SDP objective = 11

Figure 4.3: Comparison of IP and SDP relaxation

Chapter 5 Sensor Fusion in Camera Network

Camera fusion refers to the sensor fusion techniques in camera network research. In this chapter, we presents some key operations in camera fusion. The contribution of this chapter concerns mainly the *geometric fusion* problem, which aims to align the data from multiple cameras into one unified coordinate system. We also list a few commonly used *data fusion* techniques.

5.1 The Camera Fusion Framework

Traditional sensor fusion theories, although they cover a wide range of sensor varieties, lack detailed guidance concerning a specific sensor network. This section identifies the most important tasks in camera fusion and a list of key operations for completing those tasks. Our generic framework not only incorporates a variety of cameras, but it also provides a range of viable techniques for different applications.

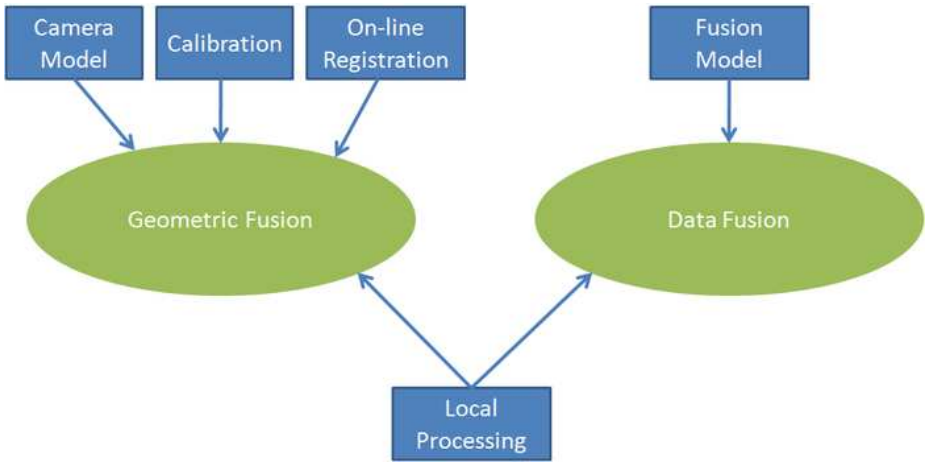


Figure 5.1: Camera fusion diagram

As shown in Figure 5.1, there are two common concerns that need to be considered when designing a camera fusion system:

1. *Geometric Fusion:* how to align the images from different cameras.

2. *Data Fusion:* once the data are aligned, how to make estimation or decision based on all the data available.

A list of key operations are need in order to answer the questions above,

1. Determining a proper *camera model* is the first step of designing a camera fusion algorithm. It has a big impact on the consequent steps, especially calibration and on line registration. The pinhole camera model is the most popular choice; however, there are a range of simplified models for specific applications, as well as a few complications to better model the cameras in real life.

2. *Calibration* is the off-line operation that estimates the position and internal parameters of all cameras in the network. The complexity of this procedure mainly depends on the chosen camera model in the application.

3. Because of the line of sight property of cameras, data alignment is not an easy task. Even for a static camera network, where the relative position between cameras can be calibrated off-line, the relationship between images obtained from different cameras also depends on the position of the object. Therefore, an *on-line registration* is usually indispensable to mapping camera views.

4. *Local processing* is conducted to reduce the communication and computation burden in a camera network. While it is important to carry out a case by

case analysis when designing the local processing algorithm for a given camera network, there are a range of image processing algorithms that are frequently applied due to their simplicity and effectiveness. These are presented in Section 5.2.4.

5. Lastly, a proper *data fusion model* is needed to make intelligent decision or estimation, based on all the data in the camera network. Traditional sensor fusion techniques are very handy in fusing image data. A lot of machine vision algorithms can also be adapted to help the fusion process, such as background modeling, object classification, object tracking and so on.

5.2 Geometric Fusion

5.2.1 Camera model

Depending on the application requirements, different camera models can be used to formulate the process of projection. The most popular model is the pinhole camera model. It successfully captures the key characteristics of the projection process and has a very convenient mathematical representation in homogeneous coordinates. However, it contains 11 parameters for each camera and it is sometimes not easy to estimate efficiently. Parallel projective models are simplifications of the pinhole camera model. Not only does it have less degrees of freedom, but it enables much more convenient mapping between camera views. I also propose a model specially designed for data alignment based on plane homography and reasonable assumptions for surveillance applications.

Pinhole Camera Model

The pinhole camera model is the most pervasively adopted model for camera networks of all types. Under this model [7], a 3×4 projection matrix is used to map a 3D point, which is represented in homogeneous coordinates as $\mathbf{X} = [x, y, z, 1]^T$ into camera coordinate $\mathbf{x} = [x', y', 1]^T$

$$\mathbf{x} = P\mathbf{X}$$

The camera projection matrix P can be decomposed into two matrices called *intrinsic* matrix K and *extrinsic* matrix $[R|\mathbf{t}]$,

$$P = K[R|\mathbf{t}]$$

The intrinsic matrix K encapsulates the camera's internal specifications as,

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

where f_x, f_y are the focal length of the camera converted into image pixel units, c_x, c_y are the center of the image plane, and the skew factor s is non-zero only when the x and y axis in image is not perpendicular, which is quite unusual.

In the extrinsic matrix, R is a 3D rotation matrix and $\mathbf{t}$ is the 3D translation vector. Together they depict the relationship between the camera center and the origin of the world coordinate system.

The pinhole camera model involves 11 degrees of freedom. Since they are coupled together during the projection process, a carefully designed calibration is required to estimate them.

Lens Distortion

Camera lenses sometimes introduce non-linearity into the projection process. If the camera suffers from significant lens distortion, an undistortion process is needed before applying the pinhole camera model.

In this research, $\mathbf{x}_d = [x_d, y_d]^T$ denotes the coordinate under distortion and $\mathbf{x}_u = [x_u, y_u]^T$ is the coordinate with the distortion corrected. $\mathbf{x}_c = [x_c, y_c]^T$ is the center of the image, and r is the distance to the image center. There are generally two types of distortion [100]:

- *Radial distortion* tries to model the radially symmetric part of lens distortion. In practice, it accommodates most of the distortion. It is usually modeled as a polynomial series with only even orders:

$$\mathbf{x}_u = \mathbf{x}_c + (1 + \sum_{i=1}^{n} kc_i r^{2i})(\mathbf{x}_d - \mathbf{x}_c)$$

- *Tangential distortion* is caused by lens being improperly aligned. Its mathematical formula is,

$$\mathbf{x}_u = \mathbf{x}_d + \begin{pmatrix} P_1(r^2 + 2(x_d - x_c)^2) + 2P_2(x_d - x_c)(y_d - y_c)(1 + \sum_{i=1}^{n} P_{i+3} r^{2i}) \\ P_2(r^2 + 2(y_d - y_c)^2) + 2P_1(x_d - x_c)(y_d - y_c)(1 + \sum_{i=1}^{n} P_{i+3} r^{2i}) \end{pmatrix}$$

Where kc_i's are a set of parameters of the radial distortion, and P_i's are parameters for the tangential distortion.

Parallel Projection Model

In order to simplify the pinhole camera model, It can be assumed that the camera center is at infinity, so that the projection lines become parallel. It is obviously an

unrealistic assumption, but it generates different levels of approximations that create mathematical convenience.

Under the parallel projection assumption, the projection matrix P becomes [7],

$$P_\infty = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & 0 \\ r_{2,1} & r_{2,2} & r_{2,3} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.1}$$

where $r_{i,j}$ are the elements of the rotation matrix R. Under this model, the intrinsic matrix remains the same as in the pinhole camera model but the extrinsic matrix is simplified, which leads to only 8 degrees of freedom.

The model in Equation (5.1) is called *affine camera model*, which is a generic parallel projection model for all the subsequent approximation models I will present. All of the following models reduce the model compolexity by applying coarser approximations to the affine camera model.

By setting the skew factor to 0, the *weak projection* model is obtained with only 7 degrees of freedom. It is a very good approximation when the average distant of the object to the camera is much larger than the distance variation of the objects. In fact, it is equivalent to first project the object onto the object plane by a set of parallel rays orthogonal to the plane, then it project the image from the object plane to the image plane by scaling the whole image by a fixed factor, which is proportional to the inverse of the average depth of the object. This process is shown in Figure 5.2.

A further simplification for weak projection model is the *scaled orthographic projection* model, which is obtained by setting $f_x = f_y$ and has only 6 degrees of freedom. The *orthographic projection* model can be derived from the scaled orthographic pro-

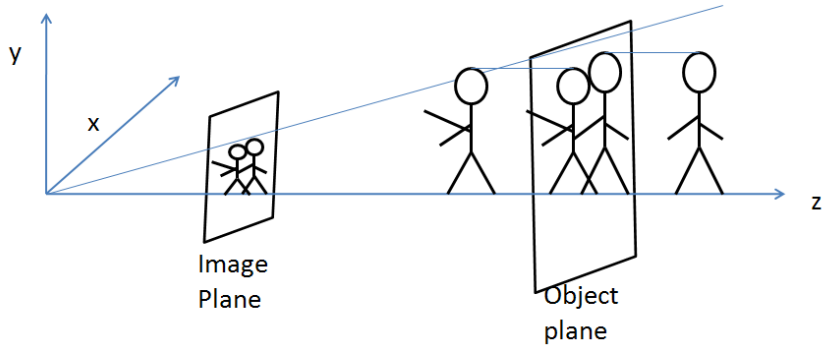


Figure 5.2: Weak Projection model. *The object is firstly projected on to the image plane by lines parallel to the image axis, then it is scaled by a factor related to the average depth of the object.*

jection model by setting $f_x = f_y = 1$.

Plane Homography Model and Blob Homography

Following the pinhole camera model, any 3D point observed in one camera can be anywhere along the epipolar line in another camera view [7, ch.9]. Therefore, there is no precise point-to-point mapping between the camera views. However, if only considering 3D points that are co-plane, there does exist a bijective mapping between the points in correspondent images between the two cameras. That is, for any correspondent points, $\mathbf{X_1}$ and $\mathbf{X_2}$, on the correspondent blobs in the two images, there is a linear mapping in the 2D homogeneous coordinate[7, ch.2] as follows,

$$\mathbf{X_1} = H\mathbf{X_2} \tag{5.2}$$

where H is a 3×3 matrix with eight degrees of freedom, known as homography matrix. Unfortunately, homography is variable with respect to the plane's depth and pose [7, ch. 13.1]. Therefore, the homography matrix has to be estimated on-line.

Using plane homography, a blob homography model is proposed, which allows us to segment the image into different foreground blobs, estimate blob to blob homography and align those blobs by applying multiple mapping for each blob.

In 3D computer vision, it is common to rectify the camera views before estimating the scene structure, so that the search for correspondent points can be significantly expedited. The rectification process finds linear mappings in homogeneous coordinate that move the epipoles of the camera pair into infinity. As a result, the pairs of conjugate epipolar lines become collinear and parallel to one of the image axes. Denoting the rectification matrices for the two cameras as H_1 and H_2, the points after rectification in two images as $\mathbf{X'_1}$ and $\mathbf{X'_2}$, and the homography between the two rectified image planes as H',

$$\begin{aligned} \mathbf{X'_1} &= H_1\mathbf{X_1} \\ \mathbf{X'_2} &= H_2\mathbf{X_2} \\ \mathbf{X'_2} &= H'\mathbf{X'_1} \end{aligned} \tag{5.3}$$

From Equation (5.2) and (5.3),

$$H = H_1^{-1}H'H_2 \tag{5.4}$$

Theorem 5.2.1 presents the form of homography matrix in rectified domain. After decomposing the homography matrix H into rectified image plane, the number of parameters that need to be estimated on line is reduced from eight to three.

Theorem 5.2.1 (Homography in rectified images) *The homography H' in rec-*

tified image domain is in the form

$$H' = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The proof of the theorem is as follows, since the homography matrix H' is up to scale, it is in the form of

$$H' = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix}$$

According to the definition of image rectification, epipoles of the two images are at infinity and in the form of $[1\ 0\ 0]^T$ and $[a\ 0\ 0]^T$. From Equation(5.3),

$$\begin{aligned} a_{11} &= a \\ a_{21} &= 0 \\ a_{31} &= 0 \end{aligned}$$

since

$$y'_2 = \frac{a_{22}y'_1 + a_{23}}{a_{32}y'_1 + 1} = y'_1$$

the following equation will always hold,

$$a_{32}y'^2_1 - (a_{22} - 1)y'_1 - a_{23} = 0$$

Therefore, all of the coefficients have to be zero. Thus $a_{32} = 0, a_{22} = 1, a_{23} = 0$. Q.E.D.

If the object in the scene is not too close to the camera, all parts of the object can be assumed to have the same depth to the camera. This is a valid assumption because in common surveillance scenarios, the foreground object is usually several

meters away from the cameras, and the depth variation of the different parts of an object is within several centimeters, which is very small portion of the overall depth. From [10, ch. 11.1.1], our constant depth assumption induces constant disparity for correspondent blobs. Based on this assumption, $x'_2 - x'_1 = d$ for each correspondent blob. Combining with the homography matrix H' in Theorem 5.2.1, we have the following,

$$(a_{11} - 1)x'_1 + a_{12}y'_1 + a_{13} - d = 0$$

which will always hold regardless of the value of x'_1, y'_1. Thus all of the coefficients for all variables must be zero, so that $a_{11} = 1$, $a_{12} = 0$ and $a_{13} = d$. The homographies between correspondent blobs depends only on one coefficient, a_{13}, which is the disparity in the rectified image.

In fact, the blob homography model essentially uses different weak perspective camera models for different blobs; it is a simplification of pinhole camera model, but a complication from the weak perspective model. The projection process is shown in Figure 5.3.

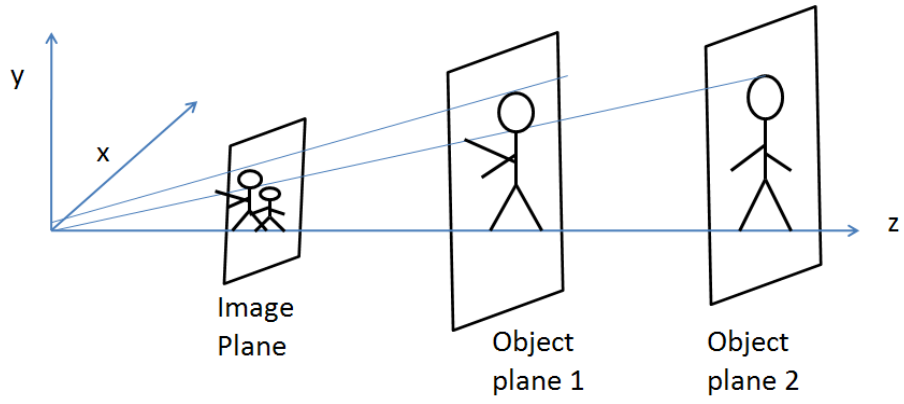


Figure 5.3: Blob Homography model. *The foreground image is firstly segmented in to blobs, then weak perspective model with different depth is applied for each blob*

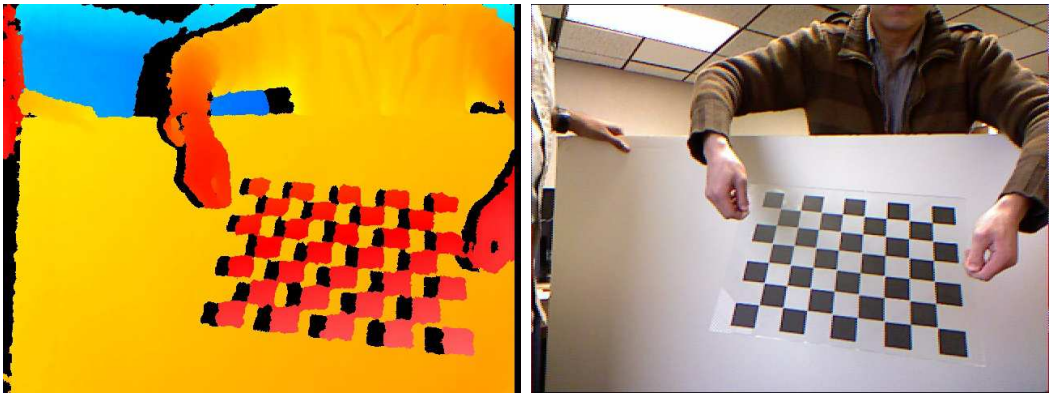
5.2.2 Camera Network Calibration

Camera calibration has been studied for decades, and a handful of robust multi-camera calibration toolboxes have emerged. I recommend [101] for estimating intrinsic parameters, including the distortion parameters for individual cameras. And the tool proposed in[102, 103] is frequently adopted for estimating the extrinsic parameters for multiple cameras in a unified world coordinate system.

However, these popular toolboxes are usually designed for homogeneous camera networks with regular visible light cameras. When cameras with different modalities are concerned, they must be adapted to handle the significant disparity between views.

For example, when dealing with a depth camera and visible camera pair, I designed a special calibration object by attaching a checkerboard pattern to a transparent glass and removing the white blocks, as shown in Figure 5.4. From the calibration the intrinsic parameters of each camera can be obtained, together with extrinsic matrix to describe the rotation and translation between the two cameras.

Another example is the calibration of a pair of thermal and visible-light cameras. Due to the phenomenological differences of objects in color and thermal images, this is not easy to implement. Our calibration is carried out by collecting the correspondent point pairs and using them to infer the geometry constraints. In fact, this is much simpler, as it bypass the explicit estimation of cameras' intrinsic and extrinsic parameters, such as focal length, aspect ratio, translation and rotation matrices, and it directly obtained the rectification matrices from correspondent points. A color tag



(a) RGB camera (b) Depth camera

Figure 5.4: Special calibration object for depth and RGB camera pair

is sticed to a round metal slice as a calibration object. As shown in Figure 5.5, the tag will be visible when heated. A color classifier using HSV color space, based on Mixture of Gaussian model, is trained to identify the tag in the video camera. Both cameras use a least square ellipse fitting algorithm to detect the center of the color tag, using them as the feature points for calibration. This is by no means the only method for calibration. Any tools that can provide unique correspondence between the two views can be used, such as an incandescent lighting bulb.

(a) video camera (b) infrared result (c) color classifier

Figure 5.5: Calibration object for thermal and regular cameras. *The pink tag in 5.5a is used as calibration object. 5.5b shows the fitting result in the thermal camera and 5.5c shows the classification and fitting result in the visible-light camera*

Once the correspondent points are collected, Hartley's method is used for rectifying uncalibrated cameras [104]. The method uses RANSAC algorithm to remove the outliers, then it utilizes the points correspondence to estimate the fundamental matrix. After finding the epipoles by decomposing the fundamental matrix, an iterative method is used to find two homograhies that map the epipoles into infinity and minimize the mapping error given by the point correspondences. Note that after rectification, the rectified images will have the same resolutions regardless of their original resolutions.

After performing the rectification, the disparity range is measured by projecting the calibration points onto the rectified domain and finding the minimum and maximum of the difference between the x coordinate of the correspondent points. This will be used to help identify the outliers during the on-line registration stage.

5.2.3 On-line Registration

Even if all 11 parameters of the pinhole camera are known via calibration, it is still not easy to align data from different camera views. This is because the mapping between camera views is a function of object depth, which is changing in a dynamic scene.

The emergence of the depth camera provides a very convenient solution to this problem. Let's consider a pair of depth and regular cameras. After obtaining the calibration data, asssuming the depth camera as the origin of the world coordinate,

$$\begin{pmatrix} x'^d \\ y'^d \\ 1 \end{pmatrix} = K^d \left[R^d \mid \mathbf{t}^d \right] \cdot \begin{pmatrix} x^d \\ y^d \\ z^d \\ 1 \end{pmatrix}$$

Where the superscribed d denotes depth camera.

Assuming the optical center of the depth camera to be the origin of the world coordinate, I can solve the equation of variable X^d, Y^d, with all of the calibration data, for each pixel with position (x^d, y^d) and its depth as the pixel value $X^d = I(x^d, y^d)$,

$$x^d = \frac{z^d(x'^d - c_x^d)}{f_x^d}$$

and

$$y^d = \frac{z^d(y'^d - c_y^d)}{f_y^d}$$

.

After obtaining the 3D coordinate of each pixel of the depth camera, they can be transformed into RGB camera's coordinate system by

$$\begin{pmatrix} x^r \\ y^r \\ z^r \end{pmatrix} = R \cdot \begin{pmatrix} x^d \\ x^d \\ x^d \end{pmatrix} + \mathbf{t}$$

where R,$\mathbf{t}$ are the extrinsic parameters for the RGB camera. Then the depth image can be aligned with the RGB image by applying the the projection matrix for the RGB camera, mapping the 3D point onto the image plane. this process can also be used to obtain the pixel appearance (R, G, B) value of each 3D point.

Of course, not all camera networks are equipped with depth camera. Alternatively, the blob homography model introduced in Section 5.2.1 can also be used to register two camera views. This method is extremely useful in a video surveillance scenarios, where the target object of the network is human body.

Under this model, for each blob, only one parameter needs to be estimated on line. This can be archived by tracking the sparse feature points between two views, such

as Kanade-Lucas-Tomasi Feature Tracker [105] and matching between views using normalized cross correlation. Since only one matching point is needed, the on line algorithm is light-weight.

5.2.4 Local Processing

In order to handle the large amount of data collected in a camera network, a lot of processing has to be conducted locally. Simple data reduction techniques, such as video/image compression, are not sufficient. In this section, I present a list of useful local processing techniques in camera network. Local processing is helpful for both geometric fusion and data fusion.

1. *Background subtraction* extracts the foreground information by subtracting the current image with a background image. It is one of the most commonly used algorithms in ridding the redundant data, while keeping the useful information. In the next chapter, background subtraction plays an important role in blob homography based geometric fusion by first segmenting the foregrounds into regions, which can be better approximated by a constant depth.

2. When the type of target object is known, *feature/object detection* can be used to directly locate the interest part of the image and only send the image containing the region of interest. Feature detection is widely applied in geometric fusion when the motion between cameras can be modeled by a restrictive model.

3. *Feature tracking* updates the information about objects based on temporal change of the images on a local camera. Since the temporal difference between

successive frames is very small, tracking on local camera can be performed very efficiently.

5.3 Data Fusion Models

This section presents three typical scenarios for data fusion in camera network, with some common algorithms.

In some applications, data fusion means that the selection of the best fit data collected from multiple cameras. Although data from multiple cameras will join the contest, only data from one camera will have an actual impact on the result. This scenario is usually called "winner takes all", which is useful when we don't want to modify the data to introduce artifacts. The selection criteria can be a distance function, a similarity metric or a likelihood function. This type of camera fusion is called *fusion by selection*.

For instance, in 3D reconstruction applications, texture information needs to be assigned for each 3D point. Sometimes multiple texture data are available for one 3D point. Simply averaging those texture data will generate a texture with an unnatural appearance. Therefore they need to be selected according to some criteria. Firstly all the texture data are kept for each 3D point. When an application requires generating a synthesis view by projecting those 3D points onto a virtual camera, the one that best aligns with the virtual viewpoint is selected. from the camera with the smallest angular distance to the viewpoint, as illustrated in Fig-5.6.

Other applications may require a certain measurement of the target. This demands an informative inference based on all the data available. The *fusion by tracking*

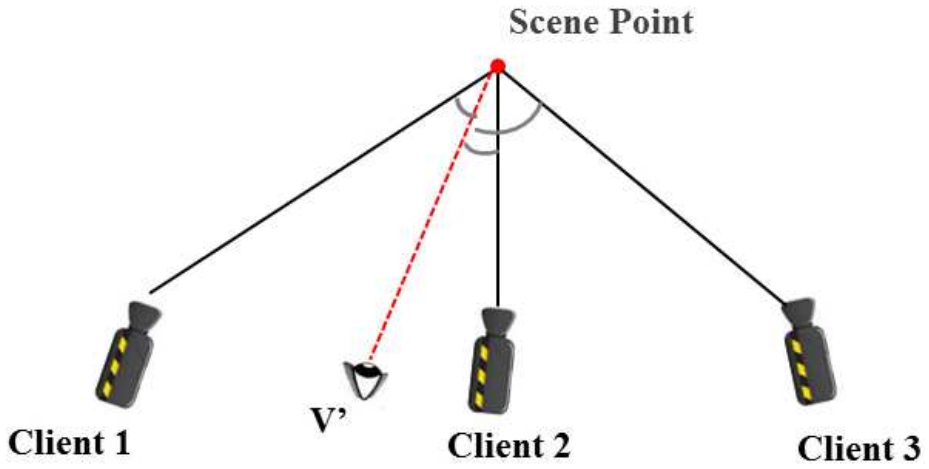


Figure 5.6: Fusion by selection. *Selection of the cameras with least angular distance for texture rendering*

scheme is one of the best methods for this application. It provides a unified probabilistic framework to fuse all the temporal and spatial information together with the prior knowledge.

At last, a lot of data fusion applications use the network data to make a joint decision, which can be called as *fusion by classification*. A lot of machine vision techniques can be used in this process, such as Background Subtraction, Support Vector Machine [106], Linear Discriminant Analysis [107], Boosting [108] and so on. The data fusion process is usually embedded by concatenating available data from all cameras into a huge feature vector. An example of fusion by classification using background subtraction techniques is presented in Section 6.1.2.

Chapter 6 Camera Fusion in Practice

This chapter presents several scenarios for camera fusion using different type of cameras. However, in heterogeneous camera networks, there are infinite number of possible combinations with different camera types, number of cameras and application specifications. This chapter focuses on fusion between visible-light cameras with thermal and depth cameras.

6.1 Human Segmentation by Fusing Visible-light and Thermal Imaginary

The first example is a system for robust segmentation of human in video sequences by fusing the visible-light and thermal imaginary. The blob homography model is used, as in Section 5.2.1. The system first performs a simple calibration procedure to rectify the two camera views without knowing the cameras' intrinsic characteristics, as mentioned in Section 5.2.2. Then a blob homography model is learned on-the-fly by estimating the disparity of each blob so that a pixel level registration can be achieved. The multi-modality information is then combined under a two-tier tracking algorithm and a unified background model to attain precise segmentation, as in Section 5.3. Preliminary experimental results shows significant improvement over existing schemes under various difficult scenarios.

6.1.1 Blob-wise Registration

Unlike most stereo vision systems, which use texture information to estimate the depth of the object, the thermal-visible camera pair does not share any similarity between the captured texture. Under the assumption of constant depth, Algorithm 7 estimates the disparity between corresponding blobs, by finding the mode of the measured disparities between a large set of corresponding pairs of pixels. The algorithm uses the contours of human blobs as the pool of correspondences and utilizes the constrains obtained from previous calibration process to boost the estimation of registration parameters.

Input: Rectification matrices H_1, H_2, disparity range $[d_{min}, d_{max}]$ and correspondent blob pairs
Output: Blob wise homography H
foreach *pair of corresponding blobs* B_1 *and* B_2 **do**
  Extract the contours of B_1, B_2;
  Rectify the contours using H_1 and H_2;
  foreach *Horizontal Scan line* **do**
    if *Both contours have same number of points* **then**
      match the points between two blobs into pairs according to the scan line order;
      Filter out the pairs with disparity out of $[d_{min}, d_{max}]$;
      Collect the disparity histogram;
  end
  if *there are enough counts in the histogram* **then**
    Get the mode of the disparity histogram $\bar{d}$;
    Obtain H' by setting $a_{13} = \bar{d}$;
    Obtain H by Equation(5.4) ;
  else
    return falure;
  end
end

Algorithm 7: *An on-line registration algorithm for thermal regular camera fusion*

Two processes are used to efficiently filter out the massive outliers. Firstly, when

a scan line have unequal number of points in two views, it is discarded without calculating any disparity. This process helps to rule out some difficult situations due to occlusion or defective segmentation. As shown in Figure 6.1, the green disparity scanned by the green line is recorded in the histogram while the red line is not counted. Secondly, disparities out of the disparity range due to false point match are simply discarded. In Figure 6.1, the rightmost points of each image alone the blue line is a false match and is likely to be filtered out by disparity range.

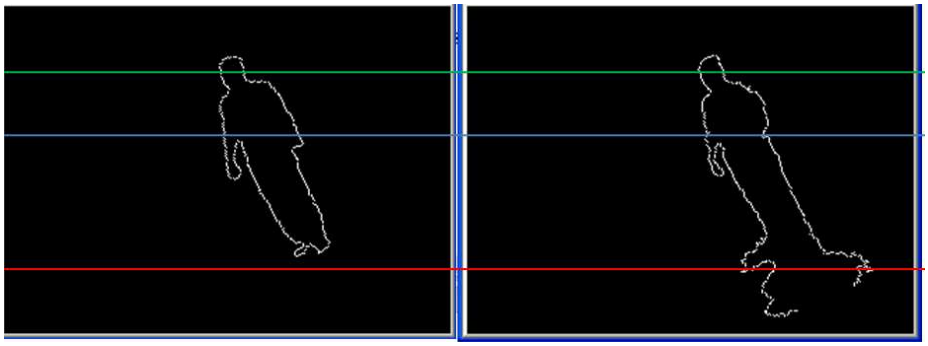


Figure 6.1: Scan line algorithm for register images from thermal and regular cameras. *Green line(upper) gives good match while red line (bottom) contains uneven number of points due to the shadow, which will be ruled out by our algorithm. Along the blue line (middle), the rightmost points of each image is a false match and is likely to be filtered out by disparity range.*

6.1.2 Robust Fusion via Tracking and Background Modeling

In reality, there can be multiple blobs in the views and due to the inaccuracy in blob segmentation, it may not be easy to find correspondent blobs between the views. Sometimes, it is simply too hard to obtain a good estimation of the blob-wise disparity due to occlusion or defective segmentation. I handle these problems by designing a two-tier tracking scheme together with a joint background subtraction.

Firstly, background subtraction is performed individually to extract the blobs from each camera. These blob information are fed into individual trackers to detect long-existing objects and filter out possible false positives. A combined tracker is then used to match objects between the two camera views, calculate and track of the disparity of each object. Using the disparity estimated from combined tracker, the homography matrix can be calculated using Equation (5.4) so that matched object can be aligned to perform a joint background subtraction. Finally, the improved segmentations from the joint background subtraction are fed back to the trackers to improve estimation of the state of each tracker. The system flow chart is shown in Figure 6.2.

Figure 6.3 is a snapshot of this process. In the previous time instant shown in Fig. 6.3a, there is only one object in each view. However, in the next time instance in Fig. 6.3b, due to the split of the shadow with the human body, the visible-light camera has two blobs after background subtraction and the individual tracker mistakenly takes the shadow blob as the new observation. In the combined tracker, there is no observation of disparity because all the point pairs are filtered out by Algorithm 7. However, thanks to temporal inferencing, the disparity estimated by the tracker is still good. By a joint background subtraction, I are able to get much better segmentation shown in Fig. 6.3c. The new information is passed back to the individual and combined trackers to improve the estimation of their states.

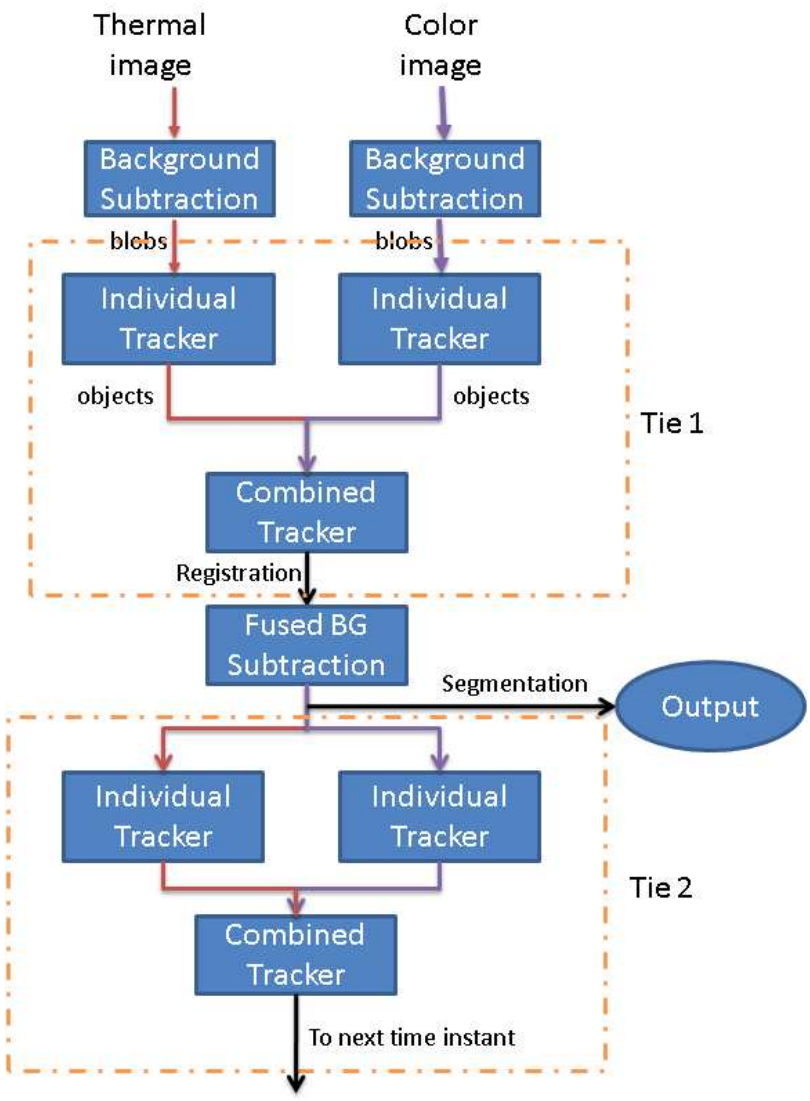
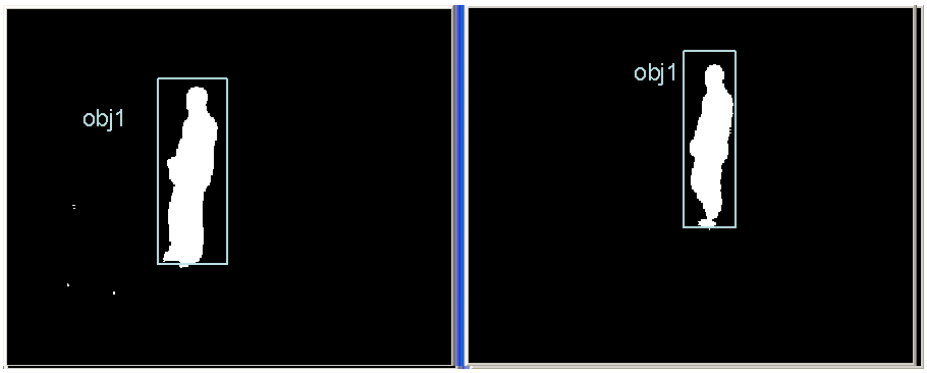


Figure 6.2: Block diagram for thermal-RGB camera fusion

Robust tracking

Each tier of the tracking process consists of simple trackers at two different levels — the individual level and combined level. The individual tracker tracks the objects' bounding box and velocity. The velocity is updated at a fixed adaption rate α.

$$v_t = \alpha v_{t-1} + (1 - \alpha)\hat{v}_t \tag{6.1}$$



(a) result in time t

(b) individual tracker at $t+1$

(c) fused result in $t+1$

Figure 6.3: Snapshot of segmentation result in successive frames. *The color bounding box shows the state in individual tracker, I see how the second tier of the tracking correct the wrong estimation from individual tracker in the first tier*

where v_{t-1} is the previous velocity, $\hat{v_t}$ is the current observed velocity. The tracker also records the number of the times the object has been observed or missed to deal with new object emergence, occlusion and noisy observation. An object will be regarded as a new object only if it has been observed more than a number of times in successive frames; an object will be deleted from the list only if it has been lost observation in a number of successive frames.

The combined tracker attempts to infer the disparity of the object using observation from both camera views, which is calculated by Algorithm 7. The state of the combined tracker is calculated by

$$z = \frac{1}{D - \bar{d}} \tag{6.2}$$

where the $\bar{d}$ is the disparity output from Algorithm 7 and D is the largest positive disparity during calibration. From [10, ch. 11.1.1] z is linearly proportional to the depth of the object to the rectified image plane. Since the observation of the disparity is much noisier than what of the individual sensor, I apply a "gating" process to rule out the apparent false estimation. If $||\hat{z}_t - z_{t-1}|| > \epsilon_1$, the observation is discarded and z is updated with $z_t = z_{t-1}$, where ϵ_1 is a design parameter. When the observation is valid, the state z is updated similar to Equation(6.1).

The two tiers of tracking basically adopt the same process. However, the result of first tier is only used to provide an estimation of the registration between two camera views. After obtaining the fused segmentation result, the state are restored to previous time instance and the second tier of tracking is used estimate the state with higher accuracy.

Background Modeling

There are three different background modeling processes in our system, two of which are performed individually in each camera view. Due to the significant temperature difference between the environment and human body, the detection of human in thermal image is relatively easy. Therefore, a static Gaussian model is used to model each pixel in the background. By collecting a fixed amount of background frames, the mean and variance (μ, σ^2) are calculated to model each background pixel. By applying this model to an incoming image, a probability map can be generated for foreground detection. The label for each pixel $\mathbf{x} = (x, y)$ in thermal image is determined by,

$$\text{foreground label } l_{\mathbf{x}} = \begin{cases} 1 & (T(\mathbf{x}) - \mu_{\mathbf{x}})^2 > \epsilon_2 \cdot \sigma_{\mathbf{x}}^2 \\ 0 & \text{otherwise.} \end{cases} \tag{6.3}$$

Where $T(x, y)$ is the pixel intensity in thermal image, ϵ_2 is a fixed threshold.

For color image, I adopt a recent non-parametric adaptive background modeling algorithm [109]. In this model, background pixel is traced in a list of code word including a color vector $\mathbf{v}_m$ and a brightness range $(\check{I}, \hat{I})$. For a test pixel vector $C_{\mathbf{x}} = (R_{\mathbf{x}}, G_{\mathbf{x}}, B_{\mathbf{x}})$, if it matched any code in the codebook, it is classified as background. A match is defined if

- Color distortion: $colorDist(C_{\mathbf{x}}, \mathbf{v_m}) \leq \epsilon_3$, where the color distortion function is defined as
$$\sqrt{||C_{\mathbf{x}}||^2 - \frac{< \mathbf{v_m}, C_{\mathbf{x}} >}{||\mathbf{v_m}||^2}}$$
where $< \cdot >$ is the inner product and ϵ_3 is a design parameter.

- Brightness: $\check{I} \leq I(\mathbf{x}) \leq \hat{I}$

When registered information from both cameras are available, I adapt the codebook background model for fused background subtraction based on two observations:

1. The infrared camera generally gives more confident classification. Specifically, in Tier 2, I can increase ϵ_2 to achieve close to zero false positive rate. Note that decreasing the false positive will decrease the detection rate as well. This is why I don't use it in Tier 1 because it will make the contour of the objects less recognizable.

2. Shadows and high lights are major source of noise in indoor surveillance for background subtraction in regular cameras, which usually have a high brightness variation but not much color distortion

Therefore, our fused background subtraction algorithm basically tightens the threshold for thermal image and uses it as a key reference and enlarges the range of each code word in order to include the shadow and high light, as shown in Algorithm 8

6.1.3 Experimental Result

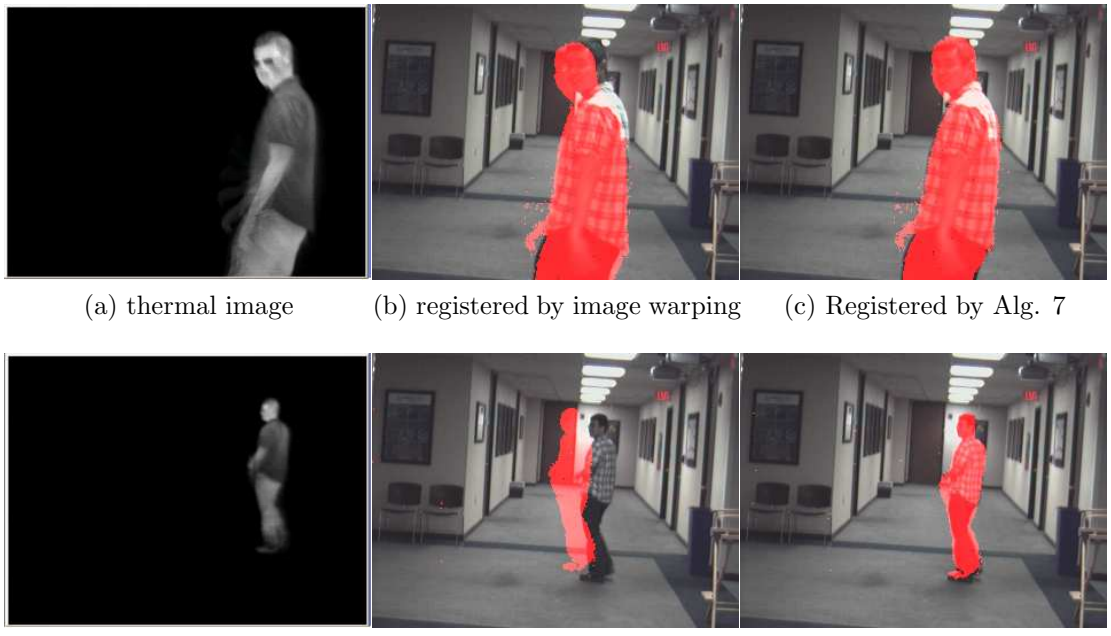
Our system consists of a Unibrain^{TM} Fire-i 400 video camera and a ElectroPhysics^{TM} PV320 thermal camera. The two cameras are fixed in a horizontal bar and put next to each other, as in Figure 6.4. The system runs on a Shuttle computer with 2GB memory and Athlon Dual core 3800+ CPU at 2.0GHz. Both cameras capture images at resolution 320×240. Our single-thread unoptimized code runs at 12.8 fps, comparing with 7.5 fps in [55].

Input: Registered thermal image T, color image $I, \mathbf{X}$, codebook background $\mathbf{v_m}, \check{I}, \hat{I}$, thermal background parameter (μ, σ), thresholding parameter ϵ_2, ϵ_3, relaxed parameter ϵ_2', $\check{I}'$, $\hat{I}'$
Output: foreground label l
foreach *pixel in video camera* **do**
  if $(T(\mathbf{x}) - \mu_{\mathbf{x}})^2 > \epsilon_2' \sigma_{\mathbf{x}}^2$ *in thermal image* **then**
    $l_{\mathbf{x}} = 1$;
  else
    $l_{\mathbf{x}} = 1$;
    foreach *code* $\mathbf{v}_m$, $\check{I}'$, $\hat{I}'$ *in the code book* **do**
      if $colorDist(C_{\mathbf{x}}, \mathbf{v_m}) \leq \epsilon_3$ *and* $\check{I}' \leq I(\mathbf{x}) \leq \hat{I}'$ **then**
        $l_{\mathbf{x}} = 0$;
        break;
    end
  end
end

Algorithm 8: *fused background subtraction algorithm*

Figure 6.4: Thermal-RGB camera fusion system setup

In the first experiment, I show that most of pervasively adopted image warping methods are not suitable for indoor surveillance application, where the depth of the target object varies in the scene. I can see from Figure 6.5, our method described in Algorithm 7 clearly outperform image warping. In the first row of Figure 6.5, I can see both methods work equally well when the calibration points are at the same depth of the object. However, when the depth of the object changed in the second row of Figure 6.5, the single homography registration in image warping is no longer accurate and the two blobs do not align. On the contrary, our registration algorithm can successfully register object regardless of its depth variation.



(a) thermal image (b) registered by image warping (c) Registered by Alg. 7

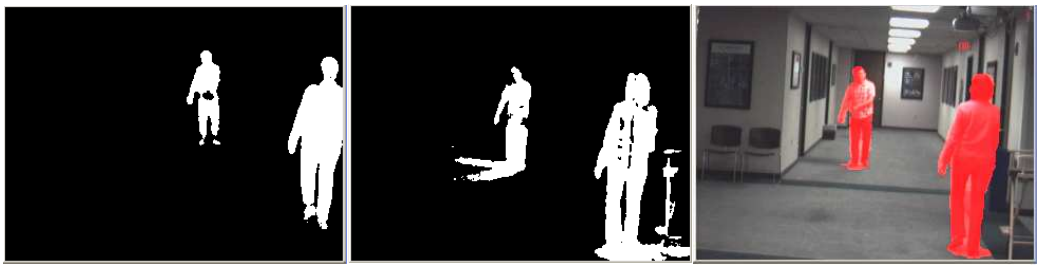
(d) thermal image (e) registered by image warping (f) Registered by Alg. 7

Figure 6.5: Registration result between thermal and regular cameras. *In the first row, I can see the registration between thermal image and regular image are both fine by using image warping and our method, shown in the red blob. However, in the second row, the image warping method fails when there is a depth variation.*

In the second experiment, I show the effectiveness of our combined tracker over

the segmentation using the two modality separately. I adopted the OPENCV[110] library for codebook implementation and use its default parameters. In background subtraction in thermal image, the threshold ϵ_2 is set to 2 to get roughly the best visual segmentation. In the combined tracker, $\epsilon_1 = 10$, $\epsilon_2' = 3$, $\check{I}' = 20$, and $\hat{I}' = 10$. Figure 6.6 is a snapshot of the tracking result. Comparing between Figure 6.6c and 6.6d, the thermal image gives much better segmentation but still has some part missing due to occlusion and low temperature appurtenance, while the code book background subtraction in visible camera suffer from illumination changes and shadows. All of these problem can be solved in the fused tracker in Figure 6.6e.

(a) thermal image (b) color image



(c) thermal background subtraction (d) color background subtraction by codebook (e) combined tracker

Figure 6.6: Comparison of our fused system with any single camera system. *Our result in 6.6e shows best results over using thermal camera alone in 6.6c or video camera alone in 6.6d*

At last, I perform quantitative measurement of our segmentation results powered

by multi-modality, multi-pass background subtraction and fusion. Twelve frames are randomly chosen and hand segmented by interactive graph cut algorithm[67]. I then compare the accuracy of segmentation by background segmentation alone and with fusion algorithm. In Figure 6.7, the three graphs in the first row are respectively image segmentation, infrared segmentation and fused segmentation. The leftmost graph in the second row is the ground truth segmentation. Then last two graphs are the image segmentation and fused segmentation results overlapped with ground truth. The pink part is the correct segment, red color denotes the false negative and the blue is the false positive. The average accuracy for the 12 frames can be seen in Table 6.1. I can see that through proper morphological operation, the fused segmentation algorithm promotes a very low false negative rate (1.6%) which is crucial for privacy protection and keep the false positive rate is at the same level as the image segmentation at the same time.

Table 6.1: Segmentation error in single camera system and fused cameras

	False Positive (type I error)	False Negative (type II error)
single camera Segmentation	0.2071	0.0965
Fused segmentation	0.1857	0.0160

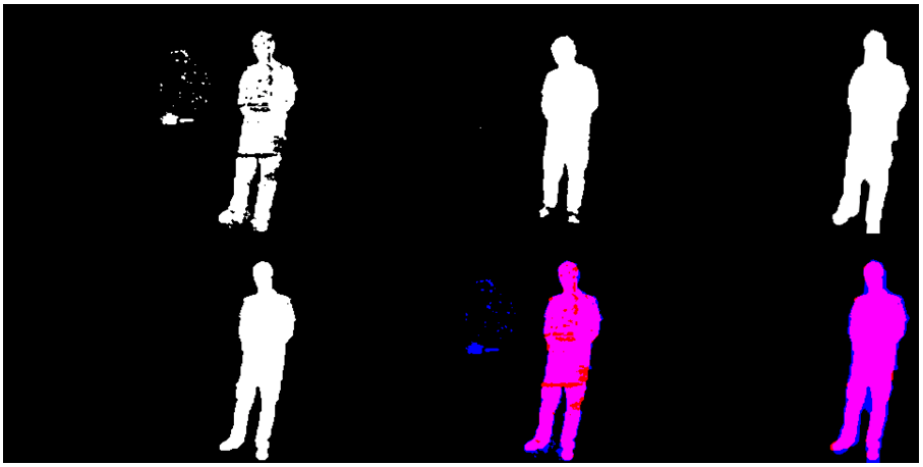


Figure 6.7: Quantitative measurement of the segmentation result. *The three graphs in the first row a) image segmentation, b) infrared segmentation and c) fused segmentation. The second row:d) ground truth, e),f) image segmentation and fused segmentation overlapped with ground truth, where pink part is the correct segment, red color denotes the false negative and the blue is the false positive*

6.2 View Dependent Rendering by Fusing Depth and Regular Cameras

Another example is proposed for real time view dependent image rendering based on 3D point clouds and color texture captured from a network of joint depth-and-color cameras, the Microsoft Kinect devices. I will see how the 3D scene is acquired and unified by a network of cameras with the help of depth information obtained by depth camera under full pinhole camera model as in Section 5.2.1 and Section 5.2.3. The texture rendering process with "winer takes all" approach is a simple example of data fusion by selection in Section 5.3. An virtual mirror system is given as an possible application of this system.

6.2.1 View Dependent Rendering System

Using the on-line registration scheme mention in Section 5.2.3, I can obtain a cloud of 3D points with its R,G,B texture information from a joint depth-and-color camera

pair, with the one of the camera center as the origin of world coordinate. In a network of depth-and-color camera pairs, I can further apply another multiple camera calibration [102] in order to unify the world coordinate system for each camera pair. Therefore, a unified 3D point cloud can be obtained with each point associate with a R,G,B value indicating its texture of the scene.

After obtaining the scene point cloud, a new view can be generated by projecting those points using a new camera projection matrix. For each pixel area on the virtual image plane, there might be zero to multiple projected 3D points. If there is only one point, that pixel will take on the color value associated with the corresponding 3D scene point. If there are more than one points, they can either be from the same 3D scene point but originated from different cameras, or they are from different 3D point with different depth but falls on the same projection line. For those belonging to the first class, their depth values would be very close to each other. For those of the second class, their depth values must be far apart and the one with the smallest one would occlude the rest. This suggest a simple procedure of first clustering all the scene points that share similar depth values and then selecting the group that is closest to the viewpoint. The clustering algorithm adopted is simply based on thresholding on the depth values. To compute the final color for the pixel, I use the scheme of *winner takes all* to select the one that best aligns with the virtual viewpoint, as I mentioned in Section 5.3

For the pixels with no display points, they need to be interpolated from neighboring pixels. A naive approach would be to perform spatial interpolation after obtaining the color values for all the pixels that contain at least one display points. I notice that

this approach creates a great deal of blending of scene objects at different depth. To better preserve object boundaries, I separate the rendering into two phases based on the depth values from the scene points – those that are at or closer than the viewer and those that are beyond. Due to out unique mirror setting, these two sets typically have very different depth values. I first start with the latter group with scene points that are far away, apply the above process of identifying color for each pixel and then perform interpolation on *both depth and color* values to fill in small gaps. These interpolated values are inserted back to the data structure of the "closer" pixels as if they are from the true 3D point clouds. In the second phase, I render all these "closer" pixels, select the correct color value based on both 3D point clouds and interpolated results, and finally perform one more round of interpolation just on the color values. Such a layered approach provides a far sharper object boundaries as it respects the inherent depth values. It is possible to increase the number of depth levels to create a better rendering but two levels are sufficient for our application.

6.2.2 Application to Virtual Mirror System

The proposed view generation system can be used to mimic a mirror in a camera-display system. As shown in Figure- 6.8(a), the physical model of a mirror is very straightforward. The mirror image is based on reflecting a light ray from the scene to our eye, which can be modeled as a pinhole mirror. Three components are necessary to reproduce this process in a virtual system:

1. Structure of the 3D scene.

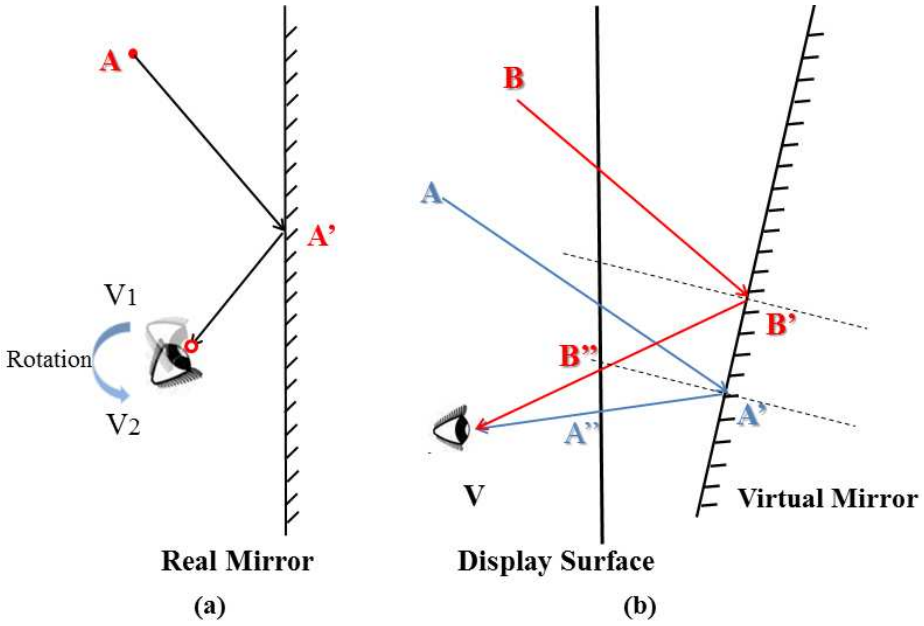


Figure 6.8: Physical and Virtual Mirror Modeling

2. 3D location of the viewpoint or more precisely, the optical center of the eye.

3. 3D location and pose of the mirror.

An important observation is that the mirror image is only determined by the location of the viewpoint but not the pose of the viewer. When the eye ball rotates from V_1 to V_2 around the same viewpoint in Fig-6.8(a), the scene point A should appear at the same spot A' on the mirror regardless of the viewing pose V_1 and V_2. As the pose of eye ball is irrelevant to the mirror image, I only need to track the position but not the pose of the viewer.

To simulate the mirror experience, I need a camera-display system that can capture the 3D world, the viewer's position and then render what a viewer should see on a virtual mirror. Notice that the virtual mirror does not need to coincide with the display. The generalized relationship between the mirror and the display is illustrated

in Fig-6.8(b): a light ray from scene point A impinging on the mirror surface at point A' gets reflected along the ray $\overrightarrow{A'V}$ towards the viewpoint V. The visual effect of the virtual mirror is presented by rendering the point A" on the display, which is the intersection between $\overrightarrow{A'V}$ and the display plane. In a similar fashion, B'' is rendered as the mirror reflection of another scene point B with respect to the same virtual mirror.

Our proposed system works much the same way as the physical model. The basic work-flow of our system is illustrated in Fig-6.9. The system first records the scene information as 3D point clouds and estimates the viewpoint position V. It then traverses each point S_i in the cloud to find the corresponding reflection point R_i on the mirror, which determines its reflection ray to hit the view point V. Once the reflection point is obtained, it computes the intersection point P_i between the display surface Π_d and $\overrightarrow{R_iV}$ on the display. A Z-buffer is used to determine if P_i is not occluded and indeed visible to the viewer. If so, the local coordinates of P_i on the display $[P_i]_d = (x_i, y_i)$ are calculated and the corresponding pixel value $I(x_i, y_i)$ is determined based on the color information stored at S_i and the viewpoint V.

Viewpoint Tracking

To provide viewpoint dependent viewing, the system needs to track the viewer's eyes' position. As our display only renders a monocular view, I track the head position rather than the actual locations of the two eyes. I approximate the head as a sphere and treat the center as our target viewpoint. While there are many high-performance sophisticated tracking algorithms in the literature, I take advantage of a single-user

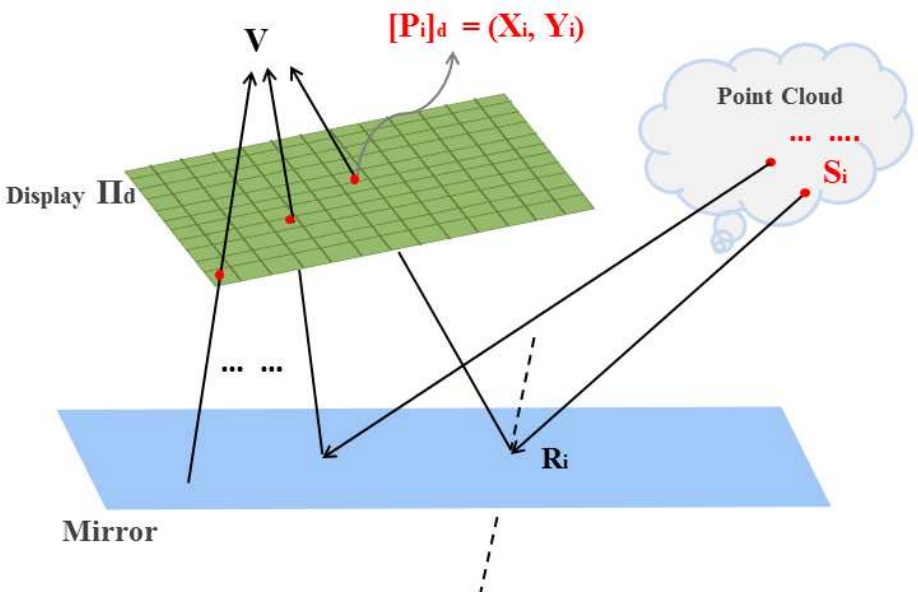


Figure 6.9: Proposed virtual mirror model

nature of the system and develop a very simple depth-based tracking algorithm. I assume that the user is much closer to the depth camera than the rest of the 3D scene points. As such, the histogram of depth values has a sharp peak of small values corresponding to the viewer that can be easily separated from the rest using a single threshold. These small depth values are then back-projected to the corresponding 2-D camera spatial coordinates. The actual depth values are not used anymore and I only keep the 2-D binary shape of the viewer. Morphological opening and closing are applied to fill in small holes and to smooth the outline of the silhouette. Starting from the topmost point of the silhouette, our algorithm follows the outline in both directions and calculates the curvature at each boundary point. The characteristic omega shape of a head induces a curvature curve that has a sharp dip from positive to negative at the two inflexion points. Detection of these two inflexion points define the extent of the head curve which are then used to fit a circle. The estimated center

of the circle on the camera plane is temporally smoothed with a Kalman filter. The actual 3D coordinates of the viewpoint is then estimated to be the 3D point that minimizes the sum of distances to each of the line formed between each camera's optical center and the center of the corresponding head circle.

Plane Mirror Image Rendering

From the virtual mirror model, I need to identify the the reflection point on the mirror followed by the display point on the display surface. However, for plane mirror, I have an alternative solution which can simplify the process and reduce the computational complexity. By making a novel adaptation of camera projection model, I can project the 3D point cloud directly onto the display plane without the time consuming reflection points finding process.

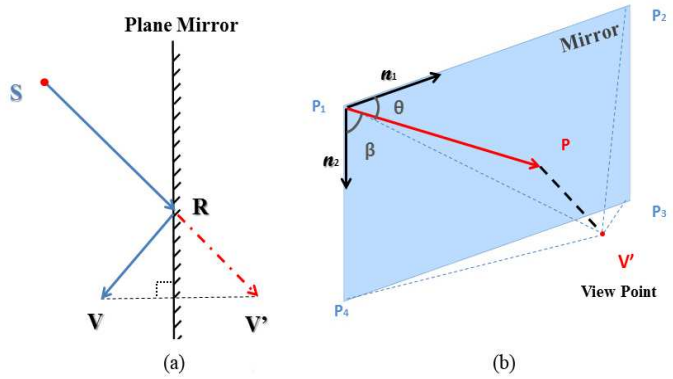


Figure 6.10: A virtual mirror rendering for planar mirror

The identification of the reflection and display points can be viewed as a process in which a scene point S is being projected on to the display plane by a virtual camera with the center of projection at virtual view point V', as seen in Figure 6.10(b). Then the whole process becomes estimating the camera projection matrix, detailed can be

found in Appendix B.

Scalable client-server architecture

I adopt a Server-Client distributed system to deal with larger rendering space and high computation complexity. Each client is responsible for one depth-color camera pair (or one Kinect) and it also contains all the calibration information.

The client generates 3D point clouds according to its color and depth images and calibration data. It also provide the first estimate of the viewpoint and the mirror image based on the previously estimated viewpoint. The interpolation procedure is postponed until each client's generated virtual image is transferred to the server. The reason of executing this operation on the server side is that the interpolation process requires the availability of all color and depth pixels from different Kinects. As all the Kinects have different views of the scene, some occluded or gap area in one Kinect could be filled in by the other clients. The distributed algorithm is summarized in Figure 6.11. The viewpoint estimation is refined and sent to all clients for the rendering of next frame.

6.2.3 Experimental Results

In this section, I present the simulation results to evaluate our virtual mirror model and analyzed the performance of the implemented system.

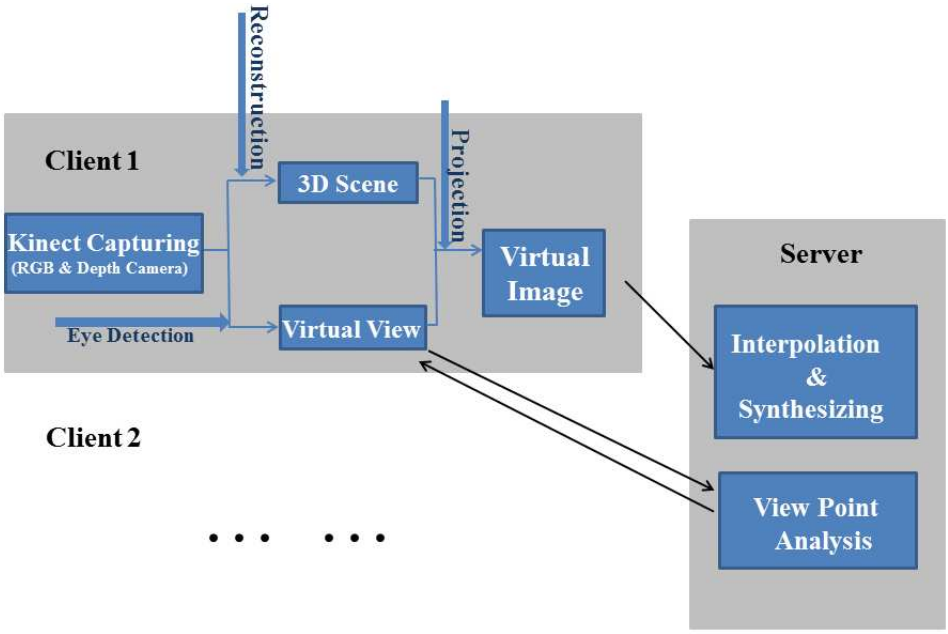


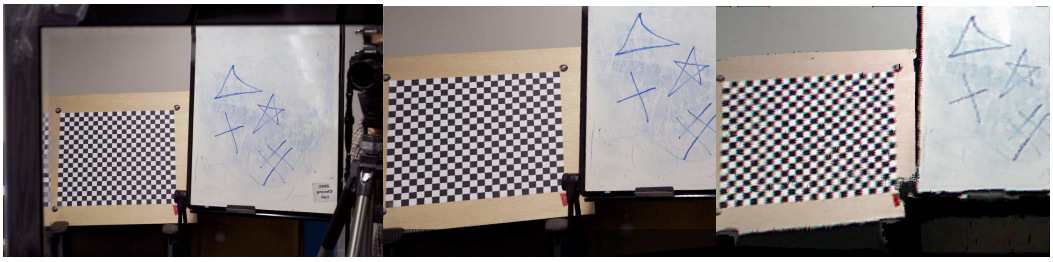
Figure 6.11: Client-Server Architecture

Rendering accuracy verification

In order to validate the accuracy of our mirror model, I made a comparison between the generated virtual image I_1 on the display and a real image I_2 taken by a digital camera looking at a real mirror which is put aligned with the display. The camera is in the same position as the asserted viewpoint that the mirror system uses for rendering.

To compare the virtual mirror image with camera captured real mirror image, I further project the captured mirror image on to the display plane by applying a homography. Here I manually choose the 4 corners of the real mirror from the taken picture to match the 4 points of generated virtual image as: $p_1 = (0\ 0)^T$, $p_2 = (0\ 1024 \times \frac{w_2}{w_1})^T$, $p_3 = (0\ 768 \times \frac{h_2}{h_1})^T$ and $p_4 = (1024 \times \frac{w_2}{w_1}\ 768 \times \frac{h_2}{h_1})^T$. w_1, h_1 and w_2, h_2 denote the size of the display and real mirror respectively. The mirror and

display are placed in the same plane with their top-left corner aligned. So with these 4 pairs of corresponding points, it is sufficient to compute homography matrix H, as seen in Figure 6.12.



(a) Real Mirror Image (b) Homography Transfor-mation on the 4 corners of the mirror (c) Generated Virtual Image

Figure 6.12: Compare our virtual mirror with a real mirror

100 matching points are manually selected on I_1 and I_2 for analysis. To reduce random noise in hand pick points, *Normalized Cross Correlation*(NCC) are applied to them: the maximum NCC value in the surrounding 3×3 pixels is chosen as matching pairs. In Fig-6.14, the similarities between these selecting points the generated virtual image and real mirror iamge are presented. They are measured by each point's $x - y$ coordinate on the image. The mean error is 1.4865 with standard deviation 0.8156.

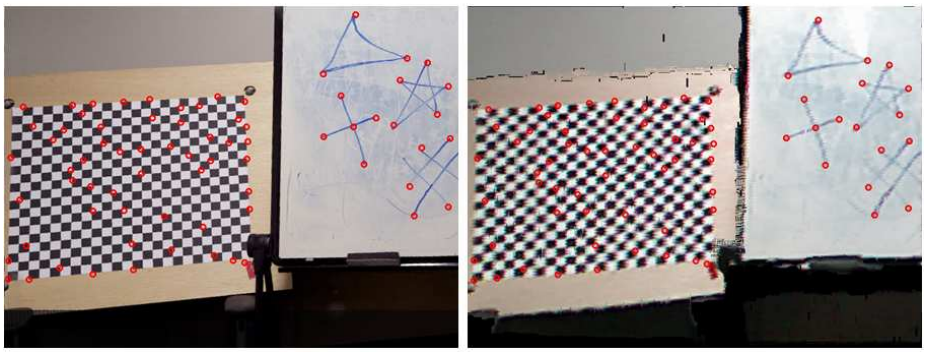


Figure 6.13: Benchmark points for rendering accuracy comparison

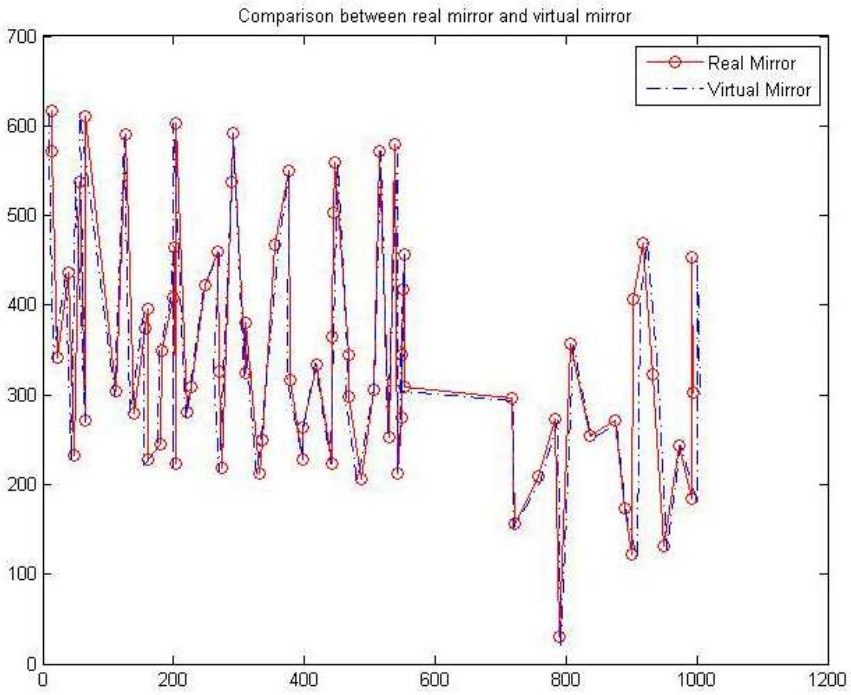


Figure 6.14: Comparison of Plotting Result

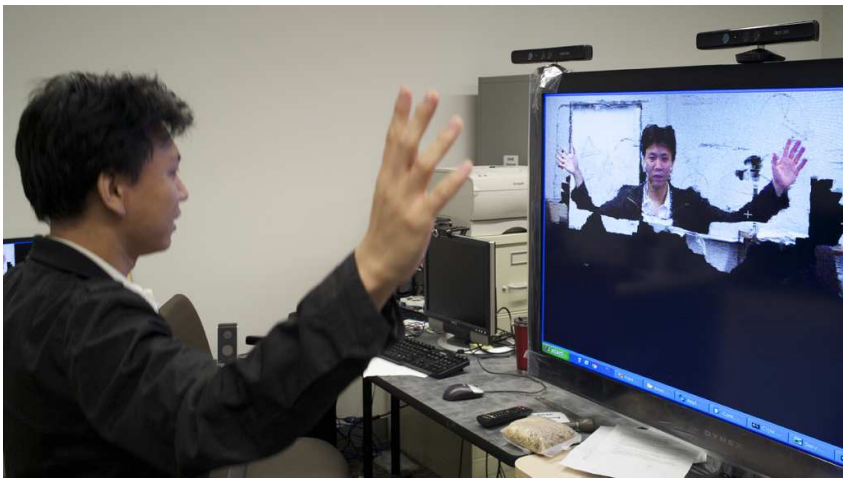
Virtual Mirror System Experiment

The mirror system was implemented using non-optimized c++ code with Intel's OpenCV library to handle image processing. Each Kinect captures 640×480 resolution video for scene points generation and the local client renders the virtual image with resolution 1024×768, which is the same size as the final image on the server. Up to this point, the system could run at frame rate 3 fps. However the speed would be greatly increased if the programe were to be adapted to the GPU. The processors details for the computers are:

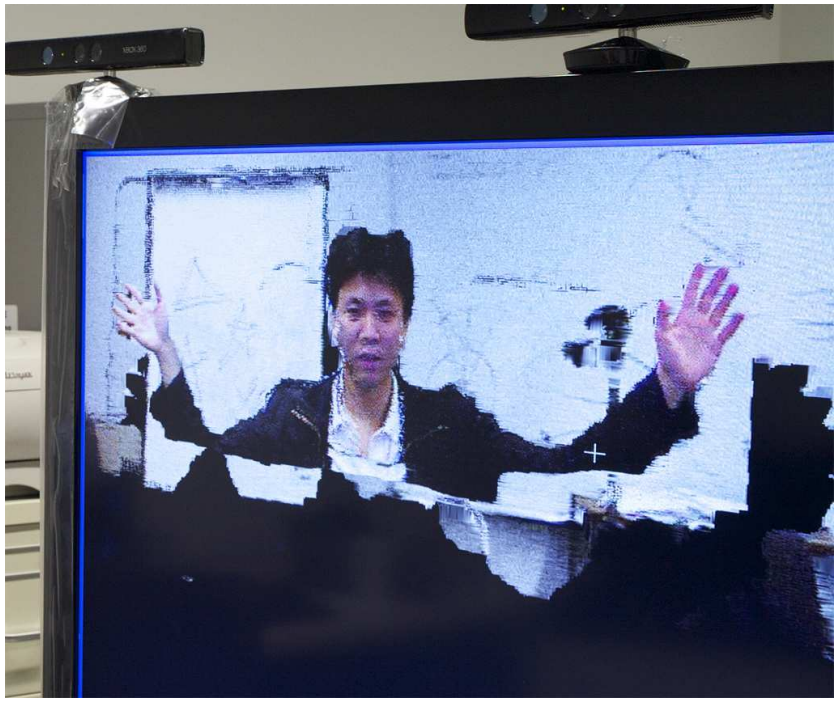
- **Server** : Intel Xeon E5335 processor with 4-core CPUs at 2.0 GHz and 4.0Gb of RAM.

- **Client** : Intel Core(TM) E8400 Duo CPU at 3.00 GHz and 8.0Gb of RAM

I show one frame of a demo video in Figure 6.15. One may notice, there are some black areas in the demo images, this because the scene is not fully covered by the Kinects. This issue could be resolved by distributing more Kinects around the display.



(a) Demo 1



(b) Capture Video 1

Figure 6.15: Virtual Mirror Demonstration

Chapter 7 Discussion

This dissertation provides significant contributions to camera network research from both camera planning and camera fusion aspects.

7.1 Discussion on the camera placement problem

First of all, in sensor planning research, a generic framework in modeling, measuring and optimizing the placement of multiple cameras is proposed. Our framework is suitable to model a variety of camera network applications for static cameras. By using a camera placement metric that captures both self and mutual occlusion in 3-D environments, we have proposed two optimal camera placement strategies that complement each other using grid based binary integer programming. Experimental results have been presented to verify our model and to show the effectiveness of our approaches. Equipped with an optimal camera placement, we have constructed a multi-camera surveillance system capable of robustly identifying and obfuscating individuals for privacy protection.

At the same time, I have presented and compared strengths and weaknesses of various well-known optimization frameworks to solve the generic camera placement problem including a greedy approach, MCMC methods, and LP and SDP relaxations.

There are many interesting issues in our proposed framework that deserve further investigation. Environmental factors such as prior knowledge of the movement of people, their inter-personal distances and configurations as well as the specifics of

the back-end vision algorithms can and should be incorporated into the models to further improve camera placement. The incorporation of models for different visual sensors, such as omnidirectional and PTZ cameras, or even non-visual sensors and other output devices such as projectors, is certainly a very interesting topic. The optimality of our greedy approach can benefit from a detailed theoretical study. The technical issues in combining wide-area calibration and visual targetging are also important problems that we believe can be overcome in the very near future. Last but not the least, the use of visual targetging in other application domains such as immersive environments and surveillance visualization should be further explored.

In addition to our simulation study on the optimization tools suitable for camera placement problem, it might be interesting to study how those algorithms can be combined together to solve the generic camera placement problem even more effectively. For example, the output of a greedy approach can be used as an initialization for the sampling methods. And while SDP relaxations outperform LP relaxation in terms of obtaining tighter bounds, it suffers from dimension increase due to the "lift and project" process. Some dimension reduction approaches may be useful to reduce the problem size and apply more layers of SDP relaxations.

7.2 Discussion on sensor fusion

Sensor fusion is yet another important topic in camera networks. Instead of applying generic sensor fusion techniques, my research focuses on how to utilize the physical camera model to improve the fusion process. Pinhole camera model is useful for studying the imaging function of most light of sight visual sensors, and it has also

a very convenient mathematical representation. Calibration is also important in the fusion process and critical in a system with stringent time constraints, where a lot of computation need to me moved into off-line process.

For example, a robust human segmentation system is presented by fusing video and thermal imaginary. After a simple calibration procedure, a blob wise registration is achieved by estimating the disparity of each correspondent blobs on the fly. The estimation of registration parameters is further improved by temporal inferencing via a two-tier tracking algorithm. The segmentation under a fused background subtraction shows significant improvement over that of using either modality alone.

Currently, the background subtraction in fused image is only a union of individual modality with tightened thresholds. Further improvement can be obtained by fusing the two modalities under a specific human body model. Also, the inference of disparity using temporal information is performed by a simple weighted averaging together with a gating process. A more sophisticated tracker, such as particle filter, may be used to estimate the disparity under a probabilistic framework. Last but not least, our system can only segment the human bodies out of the background; therefore it is interesting to see how to obtain separate segments when there is occlusion between multiple human blobs.

Depth cameras provide complementary depth information for regular visual sensors. Therefore, a mixture of depth camera and regular cameras is extremely suitable for acquiring 3D information of the scene. I propose a framework that can collect precise 3D scene information in real time, and generate a virtual mirror based on accurate physical model. The system can handle arbitrary position and orientation

of the mirror that is independent of the display monitor. Future work includes improving the efficiency of the system via parallelization and extending the reflection points localization algorithm to more generic mirror surfaces.

Chapter A Relationship between depth and disparity

As in Figure A.1, we can overlay the two rectified image planes into one plane. the baseline O_1O_2 is parallel to the plane, where O_1, O_2 are centers of the two cameras. A 3D point P is projected to the two camera through O_1, O_2 at X_1, X_2. Project O_1, O_2 to line $\overline{X_2X_1}$ at H_1 and H_2, H_1, H_2 is the zero positions in the two image planes; draw a line from P perpendicular to the image plane, intersecting the plane at H_3; from O_2, draw a line parallel to $\overline{X_2H}$, intersecting $\overline{PH_3}$ at A; Using Euclidean geometry, we have,

$$\frac{\overline{O_2O_1}}{\overline{X_2X_1}} = \frac{\overline{PO_2}}{\overline{PX_2}} = \frac{\overline{PA}}{\overline{PH_3}} \tag{A.1}$$

where $\overline{O_2O_1} = b$ is the baseline; $\overline{AH_3} = f$ is the focal length; $\overline{PA} = z$ is the depth of the point. The disparity

$$d = \overline{X_2H_2} - \overline{X_1H_1} = \overline{X_2H_2} + \overline{H_1X_1} = \overline{X_2X_1} - \overline{O_2O_1} \tag{A.2}$$

From Equation (A.1), we have, $\overline{X_2X_1} = \frac{\overline{PH_3}}{\overline{PA}} \cdot \overline{O_2O_1}$, plug it into Equation (A.2) we have

$$d = \frac{f}{z} \cdot b$$

Since b and f are constant, if we assume the depth of each point within the same blob has same depth, according to Equation (A.1), the disparity of these points also have to be constant. Further more, we can see that the inverse of disparity and the depth are linearly correlated.

$$\frac{\overline{O_2F}}{\overline{PH}} = \frac{\overline{X_2O_2}}{\overline{PX_2}} \tag{A.3}$$

Add Equation (A.1) to Equation (A.3),

$$\frac{\overline{O_1O_1}}{\overline{X_1X_2}} + \frac{\overline{O_2F}}{\overline{PH}} = 1$$

Here, ; $\overline{O_2F}$ is the focal length; Both of them are constant. If $\overline{PH}$ — the depth of the point— is also constant, the disparity $\overline{X1X2}$ has to be a constant.

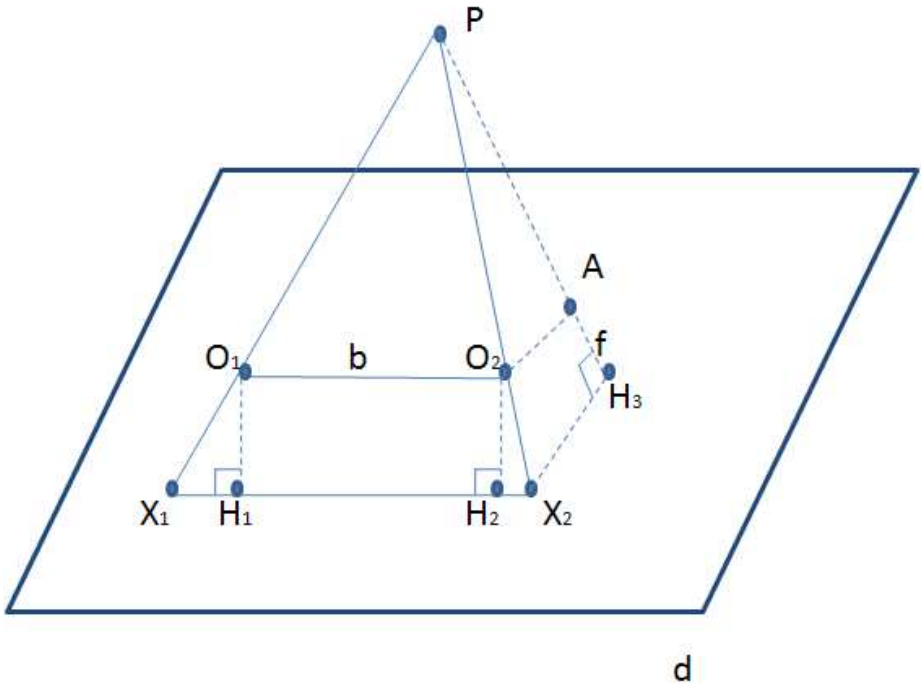


Figure A.1: *The relationship between depth and disparity in rectified image*

Chapter B Calculating the Projection Matrix for Plane Mirror

The intrinsic projection matrix K is in the form of:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

In the matrix, f_x, f_y are the focal length in pixels, which can be computed by the distance from the virtual view point to the display plane normalized by the pixel dimension. c_x and c_y is the coordinate of the projection center of view point in the image plane. To compute them, firstly we need to obtain the projection point $P(x_p, y_p, z_p)$ from V' to plane π_m:

$$\begin{aligned} x_p &= x_0 - \frac{a(ax_0 + by_0 + cz_0 + d)}{a^2 + b^2 + c^2} \\ y_p &= y_0 - \frac{b(ax_0 + by_0 + cz_0 + d)}{a^2 + b^2 + c^2} \\ z_p &= z_0 - \frac{c(ax_0 + by_0 + cz_0 + d)}{a^2 + b^2 + c^2} \end{aligned} \tag{B.1}$$

Assume the mirror has rectangular shape and the 4 corners are denoted as P_1, P_2, P_3 and P_4 (starting from top-left corner in clockwise). By convention, X and Y directions on the plane are along the two edges starting from the top-left corner P_1 as shown in Fig-6.10, then we can get the unit directional vectors:

$$\overrightarrow{n_1} = \frac{\overrightarrow{P_1P_2}}{\|P_1P_2\|} \text{ and } \overrightarrow{n_2} = \frac{\overrightarrow{P_1P_4}}{\|P_1P_4\|}$$

The offset c_x and c_y can be calculated by projecting $\overrightarrow{P_1P}$ on to the two orthogonal vectors $\overrightarrow{n_1}$ and $\overrightarrow{n_2}$, $c_x = \overrightarrow{P_1P} \cdot \overrightarrow{n_1}$ and $c_y = \overrightarrow{P_2P} \cdot \overrightarrow{n_2}$.

The extrinsic is composed of two parts: rotation R and translation $\mathbf{T}$. The role is to convert a scene point in the world coordinate into a point in the camera coordinate, which is a prerequisite to the transformation of intrinsic matrix K.

Here the rotation $\mathbf{R}$ and translation $\mathbf{T}$ are 3×3 and 3×1 matrices respectively. For details readers are referred to [7]. From the mirror plane's equation and virtual point V', it is not difficult to compute the translation matrix, as V' actually represents the offset of virtual camera center to the world origin. So $T = (V')^T$. For the rotation matrix, since we can easily computer the unit vector of image plane $\mathbf{n_i}$, we can estimate the rotation matrix which convert $\mathbf{n_i}$ into vector $\mathbf{n_z} = (0, 0, 1)$ as follows.

1. Estimate the rotation angle $\theta = \arccos(\mathbf{n_i} \cdot \mathbf{n_z})$.

2. Estimate the rotation axis $\mathbf{u} = \mathbf{n_i} \times \mathbf{n_z}$ where $\times$ stands for cross product.

3. the rotation matrix can be estimated by $I\cos\theta + \sin\theta[\mathbf{u}]_x + (1 - \cos\theta)\mathbf{u} \otimes \mathbf{u}$, where
$$[\mathbf{u}]_x = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}$$
and
$$\mathbf{u} \otimes \mathbf{u} = \begin{bmatrix} u_x^2 & u_x u_y & u_x u_z \\ u_x u_y & u_y^2 & u_y u_z \\ u_x u_z & u_y u_z & u_z^2 \end{bmatrix}$$

Bibliography

[1] Stanislava Soro and Wendi Heinzelman. A survey of visual sensor networks. *Advances in Multimedia*, 2009(640386):21, 2009.

[2] S.-C.S. Cheung, M.V. Venkatesh, J.K. Paruchuri, J. Zhao, and T. Nguyen. Protecting and managing privacy information in video surveillance systems. In Andrew Senior, editor, *Protecting Privacy in Video Surveillance*, pages 11–33. Springer London, 2009.

[3] J. Zhao and S.-C. Cheung. Human segmentation by fusing visible-light and thermal imaginary. In *International Conference on Computer Vision Workshops (ICCV Workshops)*, page 1185. IEEE, May 2009.

[4] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995.

[5] Kevin Patrick Murphy. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002. AAI3082340.

[6] Glenn Shafer. *Mathematical Theory of Evidence*. Princeton Univ Press, 1976.

[7] R. Hartley and I. Reid. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[8] J. Hecht. *Optics: light for a new age*. Macmillan/McGraw-Hill School Pub. Co., 1987.

[9] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. Artificial intelligence. MIT Press, 1993.

[10] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, August 2002.

[11] Yu-ichi Ohta, Kiyoshi Maenobu, and Toshiyuki Sakai. Obtaining surface orientation from texels under perspective projection. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, pages 746–751, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[12] Joseph L. Mundy and Andrew Zisserman, editors. *Geometric invariance in computer vision*. MIT Press, Cambridge, MA, USA, 1992.

[13] J. G. Fryer and D. C. Brown. Lens Distortion for Close-Range Photogrammetry. *Photogrammetric Engineering and Remote Sensing*, 52(1):51–58, 1986.

[14] Juyang Weng, Paul Cohen, and Marc Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14:965–980, October 1992.

[15] J. Ponce. What is a camera? *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1526–1533, 2009.

[16] Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[17] J. Urrutia. *Art Gallery And Illumination problems*. Amsterdam: Elsevier Science, 1997.

[18] T. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, 1992.

[19] V. Chvatal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory Series B*, 18:39–41, 1975.

[20] D. Lee and A. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32:276–282, 1986.

[21] D. Yang, J. Shin, A. Ercan, and L. Guibas. Sensor tasking for occupancy reasoning in a camera network. In *1st Workshop on Broadband Advanced Sensor Networks (BASENETS)*. IEEE/ICST, 2004.

[22] P.-P. Vazquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint selection using viewpoint entropy. In *Proceedings of the Vision Modeling and Visualization Conference (VMV01)*, pages 273–280. IOS Press, Amsterdam, 2001.

[23] J. Williams and W.-S. Lee. Interactive virtual simulation for multiple camera placement. In *International Workshop on Haptic Audio Visual Environments and their Applications*, pages 124–129. IEEE, 2006.

[24] S. Ram, K. R. Ramakrishnan, P. K. Atrey, V. K. Singh, and M. S. Kankanhalli. A design methodology for selection and placement of sensors in multimedia surveillance systems. In *VSSN '06: Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pages 121–130, New York, NY, USA, 2006. ACM Press.

[25] T. Bodor, A. Dremer, P. Schrater, and N. Papanikolopoulos. Optimal camera placement for automated surveillance tasks. *Journal of Intelligent and Robotic Systems*, 50:257–295, November 2007.

[26] A. Mittal and L. S. Davis. A general method for sensor planning in multi-sensor systems: Extension to random occlusion. *International Journal of Computer Vision*, 76(1):31–52, 2008.

[27] A.O. Ercan, D.B. Yang, A. El Gamal, and L. Guibas. Optimal placement and selection of camera network nodes for target localization. In *International Conference on Distributed Computing in Sensor System*, volume 4026, pages 389–404. IEEE, 2006.

[28] E. Horster and R. Lienhart. On the optimal placement of multiple visual sensors. In *VSSN '06: Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks*, pages 111–120, New York, NY, USA, 2006. ACM Press.

[29] J. Zhao and S.-C. Cheung. Optimal visual sensor planning. In *International Symposium on Circuits and Systems*, pages 165 – 168. IEEE, May 2009.

[30] J.-J. Gonzalez-Barbosa, T. Garcia-Ramirez, J. Salas, J.-B. Hurtado-Ramos, and J.-d.-J Rico-Jimenez. Optimal camera placement for total coverage. In *IEEE International Conference on Robotics and Automation ICRA '09.*, pages 844 – 848. IEEE, 2009.

[31] E. Dunn and G. Olaque. Pareto optimal camera placement for automated visual inspection. In *International Conference on Intelligent Robots and Systems*, pages 3821–3826. IEEE/RSJ, 2005.

[32] K. Chakrabarty, S. Iyengar, H. Qi, and E. Cho. Grid coverage of surveillance and target location in distributed sensor networks. *IEEE Transaction on Computers*, 51(12):1448–1453, Dec. 2002.

[33] U. M. Erdem and S. Sclaroff. Optimal placement of cameras in floorplans to satisfy task-specific and floor plan-specific coverage requirements. *Computer Vision and Image Understanding*, 103(3):156–169, September 2006.

[34] Mohammad Al Hasan, Krishna K. Ramachandran, and John E. Mitchell. Optimal placement of stereo sensors. *Optimization Letters*, 2:99–111, 2008.

[35] J. Zhao, S.-C. Cheung, and T. Nguyen. Optimal visual sensor network configuration. *Multi-Camera Networks*, ch.6:139–141, May,2009.

[36] Yi Yao, Chung-Hao Chen, Besma Abidi, David Page, Andreas Koschan, and Mongi Abidi. Can you see me now? sensor positioning for automated and persistent surveillance. *Trans. Sys. Man Cyber. Part B*, 40:101–115, February 2010.

[37] Norimichi Ukita and Takashi Matsuyama. Real-time multi-target tracking by cooperative distributed active vision agents. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 829–838, New York, NY, USA, 2002. ACM.

[38] M. Kamel and L. Hodge. A coordination mechanism for model-based multisensor planning. In *International Symposium on Intelligent Control*, pages 709–714. IEEE, 2002.

[39] Lynne E. Parker. Cooperative robotics for multi-target observation. *Intelligent Automation and Soft Computing*, 5:5–19, 1999.

[40] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[41] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4), 1992.

[42] G.L. Foresti, C. Micheloni, L. Snidaro, P. Remagnino, and T. Ellis. Active video-based surveillance systems: the low-level image and video processing techniques needed for implementation. *IEEE Signal Processing Magazine*, 22(2):25–37, 2005.

[43] M. D. Naish. Sensing-system planning for the surveillance of moving objects. *Ph.D. Thesis*, 2004.

[44] Dimitri P. Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, September 1996.

[45] Wilfried Elmenreich. *Sensor Fusion in Time-Triggered Systems*. PhD thesis, Vienna University of Technology, 2002.

[46] David L. Hall and Sonya A. H. McMullen. *Mathematical Techniques in Multisensor Data Fusion (Artech House Information Warfare Library)*. Artech House, Inc., Norwood, MA, USA, 2004.

[47] F.E. White. A model for data fusion. In *1st National Symposium on Sensor Fusion*, 1988.

[48] Alan N. Steinberg and Christopher L. Bowman. Rethinking the JDL Data Fusion Levels. In *NSSDF Conference Proceedings*. JHAPL, 2004.

[49] James Llinas, Christopher Bowman, Galina Rogova, Alan Steinberg, Ed Waltz, and Frank White. Revisiting the JDL Data Fusion Model II.

[50] Qiong Wu, Pierre Boulanger, and Walter F. Bischof. Bi-layer video segmentation with foreground and background infrared illumination. In *MM '08: Proceeding of the 16th ACM international conference on Multimedia*, pages 1025–1026, New York, NY, USA, 2008. ACM.

[51] L Volfson. Visible, night vision and ir sensor fusion. In *9th International Conference on Information Fusion*, pages 10–13: 1–4, 2006.

[52] James W. Davis and Vinay Sharma. Fusion-based background-subtraction using contour saliency. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, page 11, Washington, DC, USA, 2005. IEEE Computer Society.

[53] P. Kumar, A. Mittal, and P. Kumar. Fusion of thermal infrared and visible spectrum video for robustsurveillance. In *ICCVGIP06*, pages 528–539, 2006.

[54] Ju Han and Bir Bhanu. Fusion of color and infrared video for moving human detection. *Pattern Recogn.*, 40(6):1771–1784, 2007.

[55] P.L. St Onge and G.A. Bilodeau. Visible and infrared sensors fusion by matching feature points of foreground blobs. In *ISVC07*, pages II: 1–10, 2007.

[56] H. Cramer, U. Scheunert, and C. Wanielik. Visible, night vision and ir sensor fusion. In *Sixth International Conference of Information Fusion*, pages 1:2–10, 2003.

[57] H. Zhou, M. Taj, and Cavallaro. Target detection and tracking with heterogeneous sensors. *IEEE Journal of Selected Topics in Signal Processing*, 2(4):503–513, 2008.

[58] V. Cevher, A.C. Sankaranarayanan, J.H. McClellan, and R. Chellappa. Target tracking using a joint acoustic video system. *IEEE Transaction on Multimedia*, 9(4):715–727, 2007.

[59] Helene Torresan, Benoit Turgeon, Clemente Ibarra-Castanedo, Patrick Hebert, and Xavier P. Maldague. Advanced surveillance systems: combining video and thermal imagery for pedestrian detection. In Douglas D. Burleigh, K. Elliott Cramer, and G. Raymond Peacock, editors, *Thermosense XXVI*, volume 5405, pages 506–515. SPIE, 2004.

[60] Klaus-Dieter Kuhnert and Martin Stommel. Fusion of stereo-camera and pmd-camera data for real-time suited precise 3d environment reconstruction. In *IROS*, pages 4780–4785. IEEE, 2006.

[61] R. Reulke. Combination of distance data with high resolution images. In *ISPRS Commission V Symposium 'Image Engineering and Vision Metrology'*, 2006.

[62] Arun, Klaus Hartmann, Wolfgang Weihs, Seyed E. Ghobadi, and Arnd Sluiter. First steps in enhancing 3D vision technique using 2D/3D sensors. In OndŘ. Chum and VojtĚ. Franc, editors, *CVWW'06: Proceedings of the Computer Vision Winter Workshop 2006*, pages 82–86, Prague, Czech Republic, February 2006. Czech Society for Cybernetics and Informatics.

[63] Jiejie Zhu, Liang Wang, Ruigang Yang, James E. Davis, and Zhigeng pan. Reliability fusion of time-of-flight depth and stereo geometry for high quality depth maps. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33:1400–1414, July 2011.

[64] Y. Wang, K. Liu, Q. Hao, X. Wang, D. Lau, and L. Hassebrook. Robust active stereo vision using kullback-leibler divergence. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PP(99):1, 2011.

[65] R. Crabb, C. Tracey, A. Puranik, and J. Davis. Real-time foreground segmentation via range and color imaging. In *IEEE Computer Vision and Pattern Recognition Workshops*, pages 1–5, 2008.

[66] L. Wang, C. Zhang, R.G. Yang, and C. Zhang. Tofcut: Towards robust real-time foreground extraction using a time-of-flight camera. In *3DPVT10*, pages xx–yy, 2010.

[67] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions via graph cuts. Technical report, Cornell University, Ithaca, NY, USA, 2001.

[68] O. Arif, W. Daley, P.A. Vela, J. Teizer, and J. Stewart. Visual tracking and segmentation using time-of-flight sensor. In *IEEE International Conference on Image Processing (ICIP'10)*, pages 2241–2244, 2010.

[69] Kinect for xbox 360. http://www.xbox.com/en-US/kinect, 2009.

[70] James Newton. Become tom hanks in big with kinect's giant piano hack. http://www.kinectaku.com/news/2010/12/become_tom_hanks_in_big_with_kinects_giant_piano_hack.

[71] Vlad Savov. Kinect finally fulfills its minority report destiny. http://www.engadget.com/2010/12/09/kinect-finally-fulfills-its-minority-report-destiny-video/, December 9, 2010.

[72] Jenna Wortham. "with kinect controller, hackers take liberties". the new york times. `http://www.nytimes.com/2010/11/22/technology/22hack.html?_r=1.`, 2010, November 25.

[73] Kinect and mrpt — the mobile robot programming toolkit. `http://www.mrpt.org/Kinect_and_MRPT`, 2011.

[74] Anthony DeVincenzi, Lining Yao, Hiroshi Ishii, and Ramesh Raskar. Kinected conference: augmenting video imaging with calibrated depth and audio. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, CSCW '11, pages 621–624, New York, NY, USA, 2011. ACM.

[75] Michael Biehl, Anarta Ghosh, and Barbara Hammer. Learning vector quantization: The dynamics of winner-takes-all algorithms. *Neurocomput.*, 69:660–670, March 2006.

[76] Vlado Kitanovski and Ebroul Izquierdo. 3d tracking of facial features for augmented reality applications. In *International Workshop on Image Analysis for Multimedia Interactive Services, Delft, The Netherlands*, 2011.

[77] Cian Cullinan and Stefan Agamanolis. Reflexion: a responsive virtual mirror for interpersonal communication. In *European Conference on Computer Supported Cooperative Work*, 2003.

[78] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multi-camera people tracking with a probabilistic occupancy map. *Transactions on Pattern Analysis and Machine Intelligence*, 2007, to be published.

[79] J. Zhao and S.-C. Cheung. Multi-camera surveillance with visual tagging and generic camera placement. In *International Conference on Distributed Smart Cameras, ICDSC07*. ACM/IEEE, 2007.

[80] Brian D. Ripley. *Spatial Statistics*. Wiley, John & Sons, Incorporated, 2004.

[81] Yahya Osais, Marc St-Hilaire, and F. Richard Yu. Directional sensor placement with optimal sensing range, field of view and orientation. In *Proceedings of the 2008 IEEE International Conference on Wireless & Mobile Computing, Networking & Communication*, pages 19–24, Washington, DC, USA, 2008. IEEE Computer Society.

[82] Aaron Mavrinac, Jose L. Alarcon Herrera, and Xiang Chen. A fuzzy model for coverage evaluation of cameras and multi-camera networks. In *Proceedings of the Fourth ACM/IEEE International Conference on Distributed Smart Cameras*, ICDSC '10, pages 95–102, New York, NY, USA, 2010. ACM.

[83] Introduction to lp_solve 5.5.0.10. `http://lpsolve.sourceforge.net/5.5/`.

[84] United states census bureaustatistical abstract of the united states: 1999. United States Census Bureau, 1999.

[85] Charles M. Grinstead and Laurie J. Snell. *Introduction to Probability*. American Mathematical Society, July 1997.

[86] J. Zhao, S.-C. Cheung, and T. Nguyen. Optimal camera network configurations for visual tagging. *IEEE Journal on Selected Topics of Signal Processing*, 2(4), September 2008.

[87] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[88] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, July 1998.

[89] Rajiv Gandhi, Samir Khuller, and Aravind Srinivasan. Approximation algorithms for partial covering problems. *J. Algorithms*, 53(1):55–84, 2004.

[90] Chandra Chekuri, Kenneth L. Clarkson, and Sariel Har-Peled. On the set multi-cover problem in geometric settings. In *Proceedings of the 25th annual symposium on Computational geometry*, SCG '09, pages 341–350, New York, NY, USA, 2009. ACM.

[91] Jun S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer Publishing Company, Incorporated, 2008.

[92] Scott Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34:975–986, 1984. 10.1007/BF01009452.

[93] Bernd Gärtner. *Understanding and Using Linear Programming*. Springer Publishing Company, Incorporated, 2007.

[94] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1:166–190, 1991.

[95] Hanif D. Sherali and Warren P. Adams. A hierarchy of relaxation between the continuous and convex hull representations. *SIAM J. Discret. Math.*, 3:411–430, May 1990.

[96] J.B. Lasserre. An explicit exact sdp relaxation for nonlinear $0 - 1$ programs. *Lecture Notes in Computer Science*, pages 293–303, 2001. In K. Aardal and A.M.H. Gerards, eds.

[97] Monique Laurent. A comparison of the sherali-adams, lovász-schrijver and lasserre relaxations for 0 - 1 programming. *Mathematics of Operations Research*, 28(3):470–496, 2003.

[98] Tobias Achterberg. Scip: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1), July 2009. http://mpc.zib.de/index.php/MPC/article/view/4.

[99] Katsuki Fujisawa, Mituhiro Fukuda, Masakazu Kojima, Kazuhide Nakata, Maho Nakata, Makoto Yamashita, Katsuki Fujisawa, Mituhiro Fukuda, Kazuhiro Kobayashi, Masakazu Kojima, Kazuhide Nakata, Maho Nakata, and Makoto Yamashita. Sdpa (semidefinite programming algorithm) – user's manual. Technical report, Tokyo Institute of Technology, 1995.

[100] D.C. Brown. Decentering distortion of lenses. *Photogrammetric Engineering*, 32(3):444–462, 1966.

[101] J. Y. Bouguet. Matlab camera calibration toolbox. *IEEE Transactions on Reliability*, 2005.

[102] Tomáš Svoboda, Daniel Martinec, and Tomáš Pajdla. A convenient multicamera self-calibration for virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, 14(4):407–422, August 2005.

[103] X. Chen, J. Davis, and P. Slusallek. Wide area camera calibration using virtual calibration objects. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pages 520–527. IEEE, 2000.

[104] Richard I. Hartley. Theory and practice of projective rectification. *Int. J. Comput. Vision*, 35(2):115–127, 1999.

[105] Jianbo Shi and Carlo Tomasi. Good features to track. Technical report, Cornell University, Ithaca, NY, USA, 1993.

[106] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998. 10.1023/A:1009715923555.

[107] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.R. Mullers. Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 41 –48, aug 1999.

[108] Yoav Freund and Robert Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In Paul Vitnyi, editor, *Computational Learning Theory*, volume 904 of *Lecture Notes in Computer Science*, pages 23–37. Springer Berlin / Heidelberg, 1995.

[109] Kyungnam Kim, Thanarat H. Chalidabhongse, David Harwood, and Larry Davis. Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 11(3):172 – 185, 2005. Special Issue on Video Object Processing.

[110] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

Vita

Jian Zhao was born in Hefei, China, on November 8, 1982, the son of Jianping Gong and Ping Zhao. After completing his degree at No.1 Middle School in Hefei in 2001, he entered Zhejiang University at Hangzhou, China, receiving the degree of Bachelor of Engineer in May, 2005. He entered The Graduate School in the Department of Electrical and Computer Engineering at the University of Kentucky at Lexington, Kentucky in August, 2005. During his stay at the University, he has published 11 conference and journal papers in the field of computer vision and camera network. He is now a software developer in Microsoft Corporatoin at Redmond, Washington.